6-1-2001

Firewall strategies using network processors

Matthew Mariani

FIREWALL STRATEGIES USING NETWORK PROCESSORS

by

Matthew Enrico Mariani

A Thesis Submitted
in
Partial Fulfillment of the
Requirements for the Degree of

Masters of Science
in
Computer Engineering

Primary Advisor: _____
Dr. Yoonhee Kim, Assistant Professor, Dept. of Computer Eng.

Committee Member: _____
Dr. Roy S. Czernikowski, Professor, Dept. of Computer Eng.

Committee Member: _____
Alan Kaminsky, Research Associate, Information Technology

Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
June, 2001

Release Permission Form

Rochester Institute of Technology

Firewall Strategies Using Network Processors

I, Matthew E. Mariani, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part, on or after October 15, 2001. Any reproduction will not be for commercial use or profit.

Matthew E. Mariani

7-18-01

Date

ABSTRACT

The emergence of network processors provides a broad range of new applications, particularly in the field of network security. Firewalls have become one of the basic building blocks of implementing a network's security policy; however, the security of a firewall can potentially lead to a bottleneck in the network. Therefore, improving the performance of the firewall means also improving the performance of the protected network. With the ability to directly monitor and modify packet information at wire speeds, the network processor provides a new avenue for the pursuit of faster, more efficient firewall products. This paper describes the implementation of two simulated network processor based firewalls. The first architecture, a basic packet filtering firewall, utilizes tree-based structures for manipulating IP and transport level firewall rules while also utilizing parallelism available in the network processor during firewall rule look-ups. In the second architecture, a parallel firewall is created using a network processor based, load-balancing switch along with two network processor based firewall machines, both utilizing the basic packet filter operations of the first architecture. When added to existing routing software, these implementations demonstrate the feasibility of creating dynamic packet-filtering routers using network processor technology.

TABLE OF CONTENTS

Table of Contents i

List of Figures iii

List of Tables iv

Acknowledgements v

Glossary vi

Introduction 1
- 1.1 Purpose of Firewall Research 1
- 1.2 Organization of Thesis 2
- 1.3 Contributions 3

Background 4
- 2.1 Firewall Technology 4
  - 2.1.1 Firewall History 4
  - 2.1.2 Key Firewall Characteristics 5
  - 2.1.3 Types of Firewalls 8
    - 2.1.3.1 Packet Filtering Firewalls 8
    - 2.1.3.2 Circuit Level Firewalls 10
    - 2.1.3.3 Application Gateway Firewalls (Proxies) 11
    - 2.1.3.4 Dynamic Packet Filters 13
    - 2.1.3.5 Kernel Proxies 14
    - 2.1.3.6 Stateful Inspection Firewalls 15
- 2.2 Firewall Considerations 16
  - 2.2.1 Firewall Limitations 16
  - 2.2.2 Providing Public Services 17
- 2.3 Existing Technology 18
  - 2.3.1 Commercial Products 18
  - 2.3.2 Observations 21
- 2.4 Importance of Network Processors 22
  - 2.4.1 Emergence of Network Processors 22
  - 2.4.2 Relevance of Network Processors to Firewall Technology 23
- 2.5 Current Related Research 25
  - 2.5.1 Virus Detection 25
  - 2.5.2 Virtual Private Networks (VPNs) 27
  - 2.5.3 Intrusion Detection 28
- 2.6 Chapter Summary 29

Theory 30
- 3.1 Proposed Firewall Improvements 30
- 3.2 Basic NP Packet Filter 31
- 3.3 Parallel Packet Filter 37
- 3.4 Chapter Summary 40

Implementation 41
4.1 Development Environment 41
4.2 Basic Packet Filter 42
4.2.1 NP Hardware Tree Management 42
4.2.2 IP Firewall Rule Tree Structures Design 43
4.2.3 Transport Firewall Rule Tree Structures Design 45
4.2.4 Test Procedure 47
4.2.4.1 Software Components 47
4.2.4.2 Test Environment 49
4.3 Parallel Packet Filter 50
4.3.1 Switch Design 50
4.3.2 Firewall Node Design 52
4.3.3 Test Procedure 53
4.4 Results Analysis 54
4.5 Chapter Summary 56

Conclusion 57
5.1 Summary of Work 57
5.2 Difficulties Encountered 58
5.3 Benefits 59
5.4 Future Work 59

Bibliography 61

Appendix A – Firewall Vendors 63

Appendix B – Network Processor Vendors 64

LIST OF FIGURES

Number *Page*

Figure 1.1 – Simple Firewall Diagram 1

Figure 2.1 – Timeline of Firewall Development 5
Figure 2.2 – Back Door Vulnerability Using a Modem or Wireless Connection 6
Figure 2.3 – IPv4Packet 9
Figure 2.4 – TCP Packet Format 9
Figure 2.5 – Application Layer Firewall Architecture 12
Figure 2.6 – Protected Network with a DMZ for Public Services 18
Figure 2.7 – Sample Virtual Private Network 27

Figure 3.1 – Parallel Firewall Model 31
Figure 3.2 – Basic NP Firewall 32
Figure 3.3 – Generic NP Tree Structure 33
Figure 3.4 – Packet Filter Algorithm (IP Portion) 35
Figure 3.5 – Packet Filter Algorithm (TCP Portion) 36
Figure 3.6 – Switch-based Parallel Firewall 39

Figure 4.1 – NP System Diagram 41
Figure 4.2 – NP Tree Structure Hardware Model 43
Figure 4.3 – Transport Rule Key 46
Figure 4.4 – Basic NP Firewall Test Network 49
Figure 4.4 – Parallel Firewall Diagram 50
Figure 4.5 – Parallel Firewall Switch Algorithm Flowchart 51
Figure 4.6 – Network Switching Scheme 52
Figure 4.7 – Parallel Firewall Test Network 53

LIST OF TABLES

Number *Page*

Table 4.1 – Basic NP Firewall Network Machine Descriptions 49

ACKNOWLEDGEMENTS

The author wishes to thank the following individuals for their contributions to this thesis:

1. Jeffrey Lasky, *Director, Information Technology Lab* – Established the network processor research partnership with IBM, thus providing the network processor development tools and equipment required for the firewall implementations performed in this thesis.

2. Dr. Peter Lutz, *Assistant Professor, Department of Information Technology* – Provided design ideas for many of the aspects of the firewalls that were created, and provided highly-skilled Linux networking support

3. Louis Salcedo, *Technology Partnership Mgr, Network Processor Enablement Group, IBM* -- IBM point of contact for IBM 4GS1 and 4GS3 network processor support.

4. Bruce Hartpence, *Professor, Department of Information Technology* – Provided hardware support and information during this project.

GLOSSARY

Application Gateway: Also known as a "Proxy, " a program used to provide the user authentication and application connection mechanisms of an application layer firewall.

ASO - Advanced Software Offering: Software package developed by IBM including code for a dynamic router application on the IBM PowerNP 4GS3.

Bastion Hosts: A secured network machine or server.

Blade: One instance of an IBM PowerNP 4GS3 board. Usage example: "Packets can be transferred between multiple blades over a proprietary IBM switch fabric."

BSD – Berkley Software Distribution: Unix standard developed by the University of California, Berkley.

DMZ – Demilitarized Zone: A separate network behind a firewall that provides public access to certain network services.

FPM – Full Pattern Match: Type of search algorithm used by the IBM PowerNP 4GS3 to look-up data elements in a tree. For a given search key, the data element with the longest preceding string of matching bits will be returned.

GUI – Graphical User Interface: Interface to a application using graphical control, rather than text commands.

ICMP – Internet Control Message Protocol: Protocol used for testing connectivity between networks ("Ping").

ICT – Intelligent Connection-Tracking: Maintenance of connection state, as done by circuit-level firewalls, used by CYCON's Labyrinth firewall.

IP – Internet Protocol: Layer 3 routing protocol that is predominantly used in the Internet

LPM – Longest Prefix Match: Type of search algorithm used by the IBM PowerNP 4GS3 to look-up data elements in a tree. For a given search key, the data element with the longest preceding string of matching bits will be returned.

NAT – Network Address Translation: Process of mapping an external IP address to an internal IP address performed by a firewall or other such system. This is done in order to hide internal IP addresses from the outside world or to eliminate direct connections between internal and external hosts.

NP – Network Processor: A programmable communications integrated circuit capable of performing packet classification, packet modification, queue/policy management, and/or packet forwarding.

Packet Filter: Simple firewall that drops or "filters" packets based on simple protocol header values, including primarily IP source and destination address, network and transport layer protocol type, or TCP/UDP source and destination port.

Picocode: Term meaning IBM PowerNP 4GS3 assembly language code

Proxy: Program used to provide the user authentication mechanisms of an application layer firewall.

QoS - Quality of Service: The priority given to network traffic traveling across a router or firewall. For example, traffic from a certain host may be given a higher precedence than traffic originating from another host in order to avoid those packets being dropped, in the case of a congested network.

Security Policy: Broad term representing the applications and their associated functions put in place to provide security to a given network. For example, the rules used by a firewall are a portion of the network's security policy.

Spoofing Attack: Network attack in which the IP or TCP fields of a packet are overwritten or falsified by an intruder in order to penetrate the filtering algorithms of a firewall.

Stateful Inspection: Firewall technique of maintaining state information for existing connections across the across the firewall and filtering based on rules given for specific states.

Static Firewall: Name often used when referring to packet filtering firewalls, as the filtration rule is typically based on a fixed, existing protocol.

TCP – Transport Control Protocol: Reliable transport layer protocol providing end-to-end routing.

UDP – User Datagram Protocol: Unreliable transport layer protocol providing end-to-end routing.

VPN – Virtual Private Network: Network comprised of two physically separated networks communicating as a single, virtual network over an un-trusted network, such as the Internet. Communications between the separated networks are usefully transferred using encryption techniques over the un-trusted network.

Chapter 1

Introduction

1.1 Purpose of Firewall Research

With the expansive growth in recent years of network technology, including the Internet and e-commerce, network security has become a giant theme in the field of data communications. Large corporations and government agencies, among many others, must rely on the security of their network solutions to protect highly sensitive information and to maintain the proper function of their electronic interactions. As time has progressed, these network structures have become increasingly threatened by attacks from external entities, ranging from the robbery of secretive or private information to the exploits of hackers simply attempting to break those barriers that are supposedly unbreakable. These threats and others have led to the rapid growth of the network security industry.

Firewalls have since become the primary tool of protecting a network. Firewalls aim to protect a network by providing a single entry and exit point for network traffic, at which a security policy may be enforced. In other words, the firewall is the gateway between a trusted network and an un-trusted network, such as the Internet.

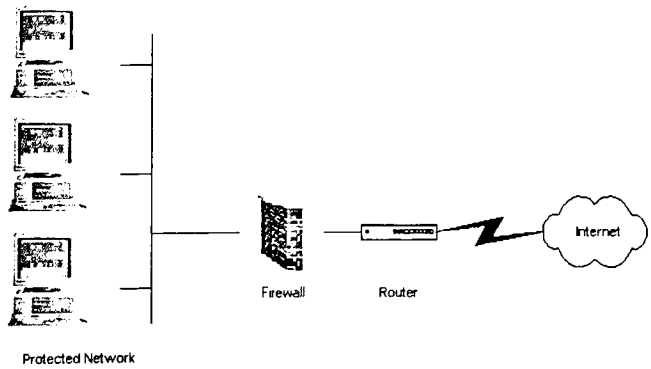


Figure 1.1 – Simple Firewall Diagram

Because the firewall is the single point of entry and exit for a network, it is clear that the network is only as secure as the firewall that protects it. In addition, as it creates a single point for concentrating network traffic, a firewall is also vulnerable to causing a bottleneck in the performance of the network. It is for these two principal reasons, security and performance, that firewall research is a necessary area of study in the field of data communications. By affecting the quality of the firewall, one directly affects the quality of the network.

1.2 Organization of Thesis

This thesis will be used to identify the shortcomings of existing firewall technology and to explore the feasibility of improving firewall technology through network processors. In order to demonstrate the abilities of network processors to enhance firewall performance, the IBM 4GS3 network processor will be used as an example platform.

A summary of the history of firewall technology will first be given in order to provide a basis for firewall design goals. Next, a discussion of the relative strengths and weaknesses of various firewall technologies will be given, along with a survey of current firewall technology. It will be shown that the weaknesses of existing firewalls relate primarily to finding a balance between quality of security and performance.

Some of the areas of research that have grown out of firewall technology will also be described, and it will be shown that network processors and the abilities they possess have not yet been explored in a research environment. This thesis explores two firewall implementations using a network processor. To begin, a basic packet filtering firewall is developed and compared in terms of performance to an existing packet filter technology. This basic packet filter is then expanded to a version involving two network processors acting in parallel in order to provide increased performance and reliability. These implementations hope to show that network processors possess the ability to allow more complex firewalls with higher efficiencies than existing technologies.

1.3 Contributions

Because firewalls continue to be both the focal point of network security implementations and the focal point of communication for a protected network, it is absolutely necessary that they possess the highest level of performance while also achieving an adequate level of security. Unfortunately, existing firewall technology tends to succeed at achieving only one of these two characteristics before the cost becomes uneconomic.

The network processor, which has become a popular theme in recent networking advancements, provides a unique ability in the firewall world to combine the speed of hardware with the diverse security functionality achievable by a software firewall solution. This thesis performs two approaches to firewall implementations in order to demonstrate the feasibility of implementing a firewall on a network processor in addition to the abilities of network processors to enhance firewall performance.

While there are many areas of research in firewall technology and network security, the network processor has yet to be fully explored. This thesis hopes to show that the network processor has the potential to become the cornerstone of network security in the future.

Chapter 2

Background

2.1 Firewall Technology

Firewall technology provides protection for the resources and information of a trusted network by allowing the controlled filtration of network traffic. This is accomplished by creating a centralized point at which a security policy may be enforced. The following sections provide some of the history of firewall technology and describe in detail the various types of firewalls that have been developed.

2.1.1 Firewall History

The firewall industry has been growing rapidly in recent years in order to meet the growing needs of businesses to protect their networks from outside attack. While growth has made the Internet security industry rather large, the industry itself is still relatively young. This section provides a brief chronology of firewall development, found primarily in [17]. The individual types of firewalls are described more specifically in section 2.1.3.

Research in firewall technology began in the mid 1980's and has led to the emergence of several types of firewalls. The first type of firewall is known as the packet filter. Papers on this very basic implementation were not released until 1988. These papers were studies published by Jeff Mogul of Digital Equipment Corporation. Shortly thereafter, AT&T Bell Laboratories (Dave Presotto and Howard Trickey) began research in a second generation of firewall called circuit level firewalls, which were based on their proprietary circuit relay technology.

This same group from Bell Laboratories was also the first to complete the development of a third generation, called application gateways. In [17], it was noted that no products or publications relating to this technology were released at the time. However, there were several private groups that began performing research on application gateways in the late 1980's, and in

the early 1990's published documents on their work. Authors of such publications included Gene Spafford of Purdue University, Bill Cheswick (also from AT&T Bell Laboratories), and Marcus Ranum. Of these, Ranum's work, involving bastion hosts running proxy services, drew the most attention and progressed into first commercial product, SEAL from Digital Equipment Corporation.

The next type of firewall to emerge, dynamic packet filters, was developed in the timeframe of 1991 by Bill Cheswick and Steve Bellovin from Bell Laboratories. Bob Braden and Annette DeSchon from USC's Information Sciences Institute further researched this technology in 1992, and in 1994 the first commercial product was released by Check Point Software.

In 1996 Scott Wiegel of Global Internet Software Group, Inc. began the development of a fifth type of firewall architecture, the Kernel Proxy architecture. This architecture was used by Cisco (the publishers of [17]) in their Centri Firewall released in 1997.

Check Point Software introduced the Stateful Inspection firewall architecture in 1999 as part of their Firewall-1 product. This architecture has formed the basis of many of today's most complex firewall systems, by combining nearly all of the characteristics of the packet filter, circuit level, and application gateway firewalls.

Below is a timeline of firewall development, modified from a diagram in [17]. Again, details of the individual classifications of firewalls are discussed in section 2.1.3.

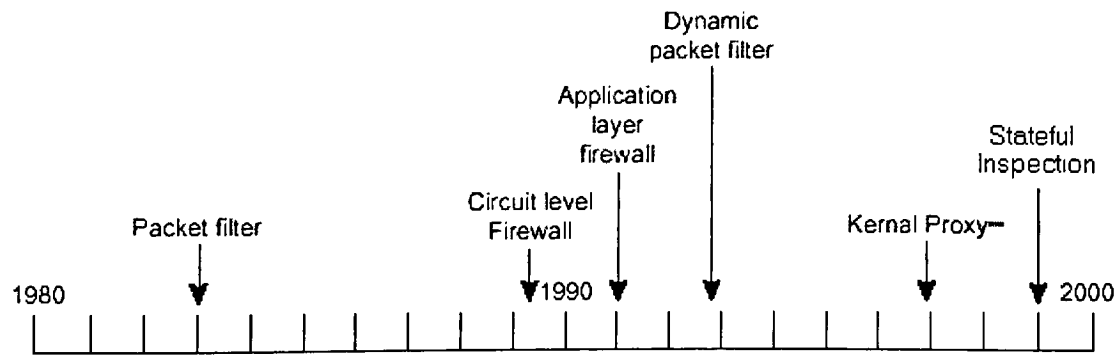


Figure 2.1 – Timeline of Firewall Development

2.1.2 Key Firewall Characteristics

Before describing the specific types of firewalls and the operations they perform, it is important to understand some of the basic properties and principal characteristics of a firewall.

As mentioned in the Introduction, a key aspect to a firewall is that it be the single point of entry and exit between the protected network and outside networks, which is illustrated in Figure 2.2. This idea is the basis of firewall design and implementation. If the firewall is not the sole point of access to the protected network, security cannot be guaranteed, and the protected network becomes highly vulnerable to means of attack, such as the so-called "back door" attack. In such situations, it is possible to bypass the firewall mechanism and connect to the protected network using alternate means, such as through a modem connection or an insecure wireless access point. This scenario is shown below.

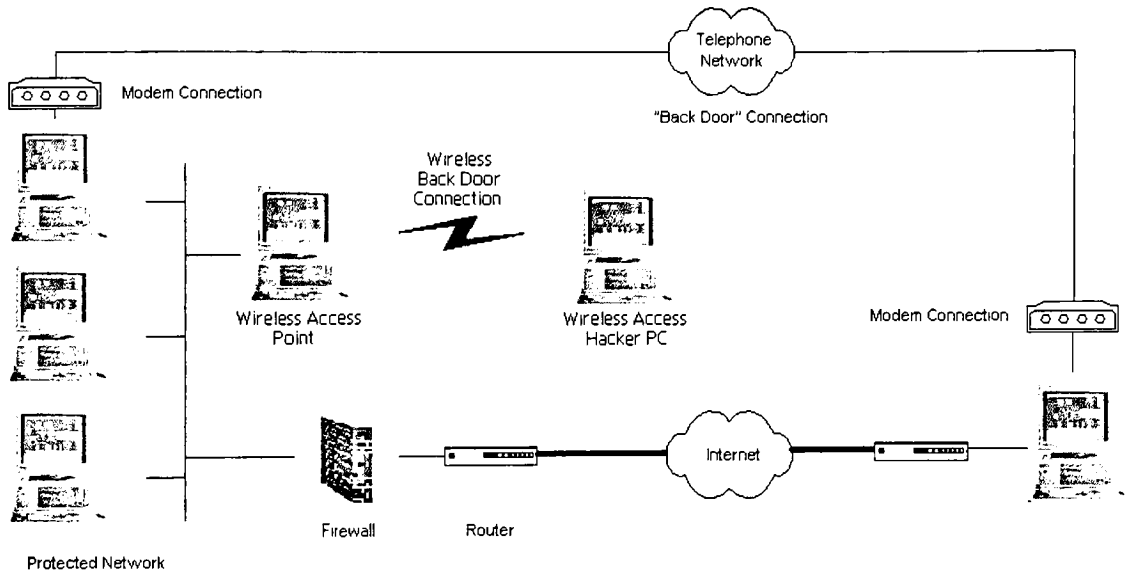


Figure 2.2 – Back Door Vulnerability Using a Modem or Wireless Connection

Another important aspect of a firewall is its security policy. The security policy is the set of rules that defines how the firewall is to permit or deny network traffic. These rules are directed at a wide variety of characteristics about the traffic and could potentially involve rules based on IP addresses, port numbers, user authentication, arbitrary connection attempts, and protocols used. It is the security policy that defines the firewall for a given network; therefore, the firewall administrator must take great care in setting up the appropriate restrictions on network traffic.

In addition to filter rules based on the properties just mentioned, firewalls have evolved to perform numerous advanced security functions, as described in [1]. For example, they may be required to maintain state information about multiple sessions accessing the secured network. This could involve the enforcement of timeout rules or monitoring the specific types of

transactions that might occur. In cases where throughput is an important concern, such as in multi-media applications, firewalls might be forced to support special low overhead protocols (i.e. UDP). Tokens or single-use passwords for user authentication may be issued by the firewall when allowing connection requests. Lastly, firewalls may even be used to perform encryption or decryption traffic in order to create virtual private networks (VPNs – see section 2.5.2) over un-trusted networks, such as the Internet.

Logging is another desirable characteristic of a firewall. It is useful for security administrators to have a record of all connections, incoming and outgoing, that occur across the firewall. Logging failed access attempts across the firewall could also be important in order to expose potential weaknesses in the firewall.

Because the firewall is the focal point for network traffic, the firewall is extremely vulnerable to becoming a bottleneck for network traffic that accesses the protected network. For this reason, it is critical that the process of validating network traffic across the firewall be as efficient as possible. The more efficient the firewall, the better performance the network will have.

The transparency of a firewall is the degree to which the actions performed by the firewall, in order to provide security for the protected network, are noticeable to the user. Transparency relates to efficiency, however differs slightly in that it embodies a larger scope of concepts. If the efficiency of the firewall is poor, users of the protected network will notice a slow-down in communication across the firewall. Obviously this implies poor transparency, as the effects of the firewall are noticeable. Transparency can also be affected by user authentication. Once a firewall has been installed, it may be necessary for a user of the protected network to authenticate with the firewall before accessing the protected network. The transparency of the firewall would be considerably low because of the added step in using the protected network. While transparency is a good thing to maintain when designing a firewall, it is certainly a double-edged sword. In general, the more security a firewall needs to provide, the less transparent it will be. Therefore, as in any design situation, a balance between security and transparency for a given firewall must be reached.

Finally, the physical implementation of the firewall is also a defining characteristic. In current technology, firewalls are implemented as either a single piece of software, or a combination of a hardware appliance and accompanying software. When the responsibilities of

a firewall are limited to simple filtering tasks such as those based on strictly IP addresses or port numbers, a hardware-based firewall is a popular choice. The simplicity of the filtering rules (security policy) and repetitive comparisons that occur lend themselves naturally to a hardware solution. Such hardware allows this type of network traffic validation to occur at higher speeds, therefore minimizing the effect of the firewall on network performance (improving transparency). Example hardware includes routers with filtering capabilities ("filtering routers") or dedicated firewall appliances. Firewall operations involving more procedural activities such as user authentication are most often implemented in software, as an application running on top of an operating system. It should be noted that operating systems might also be modified to perform packet-filtering operations; however, such an implementation will be noticeably slower than hardware systems. Several specific examples of both hardware and software solutions will be discussed in section 2.3.1. Because most current firewall implementations require simple packet filtering, in addition to complex authentication operations, systems combining filtering routers and OS firewall applications are often employed.

No matter the physical devices being used, the purpose of a firewall is to make a protected network as secure as possible by concentrating its efforts at a single point. There are a variety of operations a firewall may perform; however, it is most important to properly match the capabilities of the firewall with security needs of the organization or network being protected.

2.1.3 Types of Firewalls

In section 2.1.1, a timeline of firewall development was presented. Packet filtering firewalls, circuit level firewalls, and application gateways are considered the fundamental forms of existing firewalls, while dynamic packet filters, kernel proxy, and stateful inspection firewalls are considered to be conglomerations of the original three. This section describes each type in further detail.

2.1.3.1 Packet Filtering Firewalls

Packet filters are the most basic type of firewall. In this type of firewall, each incoming network packet is compared against defined rules based on low-level protocol fields. Depending on the rule and the contents of a packet, it is either dropped or forwarded by the firewall. For example, the Internet uses IP for routing and most often uses TCP for transport

layer functionality. The format of these protocols is shown on the following page. Most existing packet filters use combinations of the IP source and destination address and TCP source and destination port with verification of the transport layer protocol being used. Naturally, other fields in any of the protocols may be used. As an example, a firewall

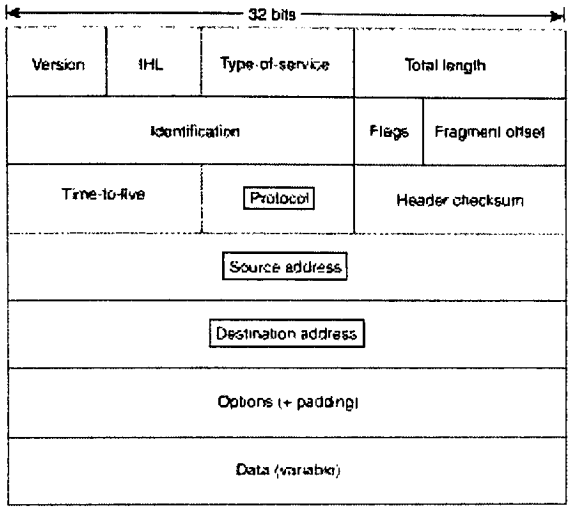


Figure 2.3 – IPv4 Packet

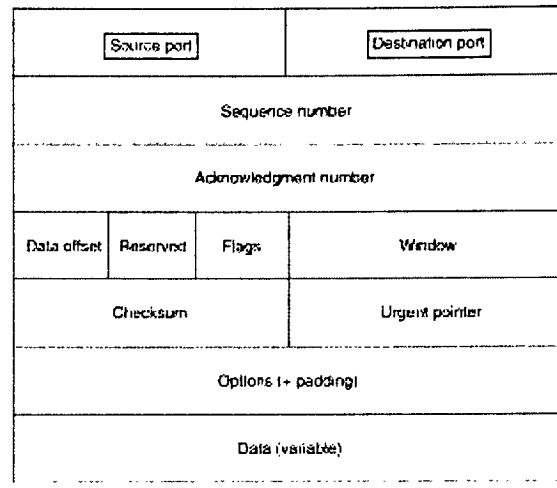


Figure 2.4 – TCP Packet Format

could accept all TCP and UDP traffic, but deny all traffic using some alternate transport layer protocol. In this case, the foreign protocol could represent an attack on the protected network.

It is relatively clear that in order to setup such a firewall, it is necessary to have detailed knowledge of the protocols being used within the protected network. The administrator may modify the specific field values being monitored by the firewall, but the fields utilized by the firewall are obviously fixed to the protocol being used. For this reason, packet filters are often referred to as static firewalls.

Packet filters may handle network traffic in one of two ways. First, all packets could be accepted unless explicitly denied by the firewall rule, or secondly, all packets could be denied unless explicitly permitted. The first version is appropriate when the level of network security is relatively low, such as in college campus networks, because access to the network is usually granted by default. The second would be useful where security is more essential, as in a government intelligence agency or corporate network.

The advantages of packet filtering firewalls are as follows. Because packet filters are based on simple comparisons between the filter rules and the protocol field values of network traffic,

they tend to be rather simple to implement. This simplicity allows packet filters to provide a fast, efficient, and generally inexpensive solution. For this reason, packet filters have been integrated at a hardware level into routers (as mentioned before, these are called filtering routers or choking routers). Packet filters also have the benefits of being able to hide internal IP addresses from external networks and avoiding the need to configure internal hosts individually. All the "work" is done by the firewall.

On the negative side, the static nature of packet filters makes them rather un-adaptable to the introduction of new protocols. In many cases, adding a new protocol to the network means direct modification of the existing firewall, which can often be a costly expenditure, especially if a hardware firewall solution has been used. Next, because packet filters simply make brainless comparisons, they are highly vulnerable to "spoofing" attacks. In spoofing attacks, protocol fields such as the IP source and destination address are falsified, in order to break through the firewall. It is a trivial task for a hacker to perform such an attack. This weakness exists partly because packet filters are not able to perform user authentication. On a higher level, packet filters do not possess the ability to examine or to modify the content of a packet. This leaves them vulnerable to content-related problems, such as the transmission of viruses, application layer violations, etc. Lastly, packet filters do not maintain any state or session information after connections occur across the firewall. In summary, packet filters have the ability to regulate where a packet may travel (i.e. across the firewall), but do not have the ability to affect or to filter based upon the application level data that might be communicated.

2.1.3.2 Circuit Level Firewalls

The next type of firewall, the circuit level firewall, provides security at a slightly higher level than packet filters, by preventing direct connections between internal and external machines [14]. Circuit level firewalls have the ability to determine whether a packet is a connection request or is part of an existing, validated connection.

The key feature of this type of firewall that differs from the packet filter is that circuit level firewalls maintain a small amount of state information, in the form of a table of all connections between internal and external networks. When a connection request arrives, the firewall adds the request to its table and tracks the state of the connection through the handshaking steps required by the transport layer protocol until the connection (or circuit) is established. In the

Internet, this is usually TCP. As each packet arrives, the firewall searches its connection table for a matching session. If one is found and is validated, the packet may pass across the firewall. If a match is not found, the packet is dropped. In order to perform the appropriate checks, the firewall could maintain session information including a session identifier unique for each connection, the specific state of the connection (i.e. handshaking, etc), sequencing information, source and destination IP address, and/or the physical interface through which the connection is established [17]. Because circuit level firewalls will only permit the transmission of a packet if the associated connection is in a valid state, the actions of circuit level firewalls are at very low-level compared to true stateful inspection firewalls (see section 2.1.3.6).

To further protect against spoofing attacks, circuit level firewalls may even modify the destination addresses of packets in order to make the packets appear to have originated from the firewall host, rather than the actual source machine. This technique is known as network address translation (NAT).

Circuit level firewalls have several advantages and disadvantages. The ability to maintain state information about connection requests is a substantial benefit in protecting a network against spoofing attacks. Also, because circuit level firewalls work at a relatively low level of the protocol stack, they still possess relatively fast performance. On the other hand, like packet filters, this type of firewall is tied permanently to the transport protocol used by the network (i.e. TCP). Also, circuit level firewalls also lack authentication mechanisms and the ability to filter based on application layer criteria.

2.1.3.3 Application Gateway Firewalls (Proxies)

Application layer firewalls, also called application gateways, perform filtering based on application layer protocols rather than at the network or transport layer, thus giving the firewall the ability to perform user authentication including username and password validation. Like the firewalls described so far, the application gate resides between a protected network and the outside world. Most application gateways operate by running a special purpose program designed to perform the security functions of the firewall. These special purpose programs are often called "proxies" or "proxy servers." Proxy servers are written specifically for each application protocol being used. For example, a protected network running telnet and ftp would need individual telnet and ftp proxies.

When a user wants to communicate with another system across the proxy, it is the proxy that performs the user authentication and security checks (usually through username and password), rather than the actual client. It is a goal for this authentication to be transparent to the user, so that the user observes direct communication with the client service. Like circuit level firewalls, no direct connections are actually made between the two systems. Because of this, application layer firewalls may perform most of the same security checks that are performed by circuit level firewalls, including checks against address spoofing. Application gateways are also capable of performing network address translation. A diagram of the proxy architecture from [17] is shown below.

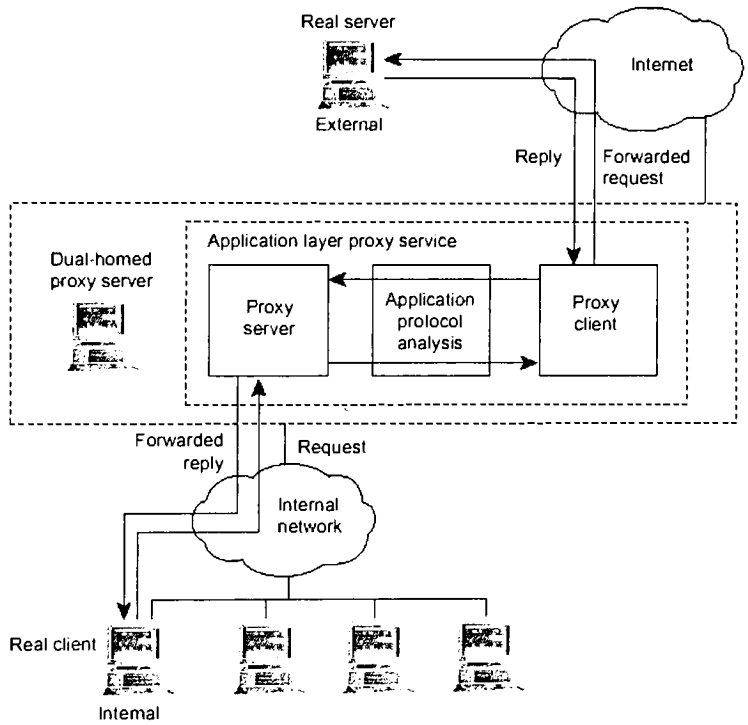


Figure 2.5 – Application Layer Firewall Architecture

Application gateways are implemented as programs running on top of a host operating system. Thus, each packet must pass up the entire network protocol stack and then pass into the application domain of the proxy system before being inspected. If the packet is validated, it must then pass back down the protocol stack for re-transmission. Because of this necessary, intermediate series of steps, application layer firewalls are comparatively slow solutions.

Proxy servers are beneficial because they are able to understand and enforce higher-level protocols that packet filters and circuit level firewalls cannot. Therefore, proxies are capable of

performing a large variety of security operations, in addition to those of more primitive firewalls. In addition to items such as user authentication, they are well suited to perform such things as connection logs and NAT. While the list of benefits for proxies is rather extensive, the list of negatives is equally long. To name a few, running a proxy on a server often requires modifications to the native network stack. Plus, proxies are vulnerable if a hacker can find a hole in low-level protocols that do not reach the application layer. Next, a unique proxy must be written for each protected application and each new protocol, making scalability a poor quality. Proxies listen on the connection port of the application, thus you cannot run application servers on the firewall server. Most importantly, proxies add a significant performance delay to the network. In order to balance the authentication abilities of proxies and the speed of packet filters, it is no surprise that combinational firewall systems are typically the norm.

2.1.3.4 Dynamic Packet Filters

This next type of firewall, the dynamic packet filter, is a more recent type of firewall technology. Dynamic packet filters combine basic packet filtering with the creation of dynamic stateful rules for connections across the firewall.

As with packet filters, this type of firewall ensures that all network traffic conforms to the network security policy by validating the IP source/destination addresses and the TCP port numbers of existing connections. When a connection request occurs, the firewall allows a limited amount of time for the receiver to respond. Assuming a response occurs with the appropriate IP and TCP port information, the firewall may deduce the existence of a host on an un-trusted network. The dynamic packet filter then begins storing limited state information, similar to a circuit-level firewall. However, these connections are established with stateful rules that may be modified on the fly (dynamically). Each connection is unique by definition, and the dynamic rules reflect the state of the connection at any moment in time. This occurs even if multiple connections exist between the same two hosts, and the connection is immediately invalidated if the receiving host does not respond within the allotted portion of time. All network traffic associated with a given connection is validated by an application running on the host machine, which allows for more extensive security checks relating to the connection.

Therefore, this type of firewall combines some of the characteristics of both low-level and application layer firewalls.

Dynamic packet filters have the greatest significance when used with applications that utilize transport protocols for limited information requests. Typically this is a stateless protocol like UDP or ICMP. In general, dynamic packet filters are less secure than application layer firewalls (there is no user authentication); however, they are more secure than circuit-level firewalls and packet filters. Performance characteristics place this firewall between application layer firewalls and lower-level types.

2.1.3.5 Kernel Proxies

The next type of firewall, the kernel proxy, is another more recent approach at firewall implementation. This type of firewall was found in [17], and was introduced as part of Cisco's Centri Firewall product.

As the name implies, kernel proxies provide proxy security services; however, these services are integrated into the kernel rather than operating at the application level. The goal of a kernel proxy is to optimize the efficiency of the firewall by handling security functions at only the levels of the protocol stack that are applicable. This can be illustrated using some of the previously discussed firewalls. Packet filters, as mentioned, perform simple comparisons on network and transport (i.e. IP and TCP) layer fields. From an operating system perspective, these fields are handled relatively low in the network stack. On the other hand, the user authentication that must be performed in application level firewalls requires packet information to be passed up the entire protocol stack into the application domain for processing by the proxy server. In other words, different firewalls process information at different levels in the protocol stack.

The kernel proxy methodology attempts to combine the security features of both low level firewalls (packet filters) and high level firewalls (application gateways) by executing the security services within the kernel space and using the application level services only necessary. When a packet arrives, the information is passed up the stack of appropriate kernel proxies. At each layer, the appropriate proxy service performs its security checks and passes the packet to the next layer if and only if necessary (meaning the packet is valid at the current layer). Each layer's proxies may maintain state information to make security decisions. For example, the Cisco Centri implementation used a set of eight kernel proxies [17]. An IP proxy handles source and

destination address checks. Another proxy exists for TCP variables. Higher-level proxies also exist for activities such as FTP and telnet, and could forward user authentication requests to higher-level extension applications as necessary. An incoming FTP request would be passed through the IP, TCP, and FTP kernel proxies only, rather than the entire protocol stack.

By implementing these proxy services within the kernel, this type of firewall is able to increase the performance of the firewall by handling security checks exactly when they should be performed. Kernel proxies help avoid the case where invalid network packets unnecessarily reach the application layer for processing.

2.1.3.6 Stateful Inspection Firewalls

Similar to kernel proxies, the stateful inspection firewall is a hybrid of the three basic types of firewalls: packet filters, circuit level firewalls, and application gateways. Like packet filters, this type of firewall is able to filter packets at the network layer. Using knowledge of the application level protocols, the firewall is able to maintain state information on all existing connections at the application level. The firewall can then make decisions whether or not to permit certain communication or operations based on the current connection status. For example, decisions could be made on user privileges or the connection source and destination.

Most importantly, maintaining state through the observation of application content allows the firewall to permit uninterrupted, direct connections between a source and destination system. While proxy services (application gateway firewall) require the user to be authenticated by the firewall, the stateful inspection system makes the firewall transparent to end-users by recording the state of the session (including user authentication) between the source and destination machine. In addition, stateful inspection eliminates the need to run a separate service (proxy) for each application level protocol.

In summary, stateful inspection firewalls are able to provide the security features of all three basic firewall types, while adding a level of transparency not found in typical application gateways. Naturally, offering this high level of security along with transparency makes stateful inspection systems highly complex and often very expensive.

2.2 Firewall Considerations

As long as they are the single point of entry to a protected network, firewalls are effective means of providing security. However, firewalls are not perfect in all respects. This section summarizes the limitations and shortcomings of firewall technology.

2.2.1 Firewall Limitations

The bottleneck problem, which has already been introduced, is perhaps the key limitation to using a firewall for network security. The apparent speed of the protected network is directly related to the quality of the firewall. As the complexity, transparency, and scalability increase, the performance decreases. It should be noted that while the performance impact of the firewall is a disadvantage, the bottleneck for network traffic is necessary in order to centralize the application of the network's security policy.

The simple firewall architectures, packet filters and circuit level firewalls, exhibit the highest amount of performance and efficiency, but sacrifice a variety of desirable characteristics in order to maintain high filtering speeds. To begin, these firewall types are highly vulnerable to so-called "content-related" attacks. Packet filters and circuit level firewalls lack the ability to look into the content carried the application layer protocols of a packet. Thus, the application layer can be used as a carrier for application attacks or the transfer of viruses. If an attacker is able to penetrate the firewall, through techniques such as "IP spoofing," the protected network is completely vulnerable to malicious traffic contained in the application layer of the packets. A second shortcoming of simplistic firewalls is the ability to perform user authentication. The ability to perform user authentication also requires the firewall to look into application level information, an ability that simple filters do not possess, as just mentioned. The ability to maintain detailed state information is yet another characteristic that packet filters and circuit level firewalls do not possess. Maintaining the state of an application level connection across the firewall ensures that valid operations are occurring throughout the duration of the connection. A final drawback of packet filters is inability to modify the content of packets passing through the firewall. This would allow the firewall to perform techniques such as address redirection and network address translation. All of these shortcomings must be considered when using simple filtering firewalls. If application level filtration is required, a more complicated, although slower, firewall will be required.

There are also a variety of limitations that are applicable to firewall technologies in general. One inherent limitation is the dependence on given protocols. Any security policy must be defined with respect to a specific set of protocols. For example, filtering firewalls are typically implemented to filter against IP or IPX at the network layer and TCP or UDP at the transport layer. Similarly, higher-level firewalls depend on the format of the application level protocols (i.e. FTP and telnet). Introducing a new protocol or application to a network usually requires a modification to the firewall. This process is often a difficult job, especially for simple, low level filtering architectures, where scalability is often not considered. Only a small amount of modern firewall products allow the ability to easily add new or custom protocols to the protected network.

Another inherent dependency, especially in the case of application level firewalls, is a reliance on the underlying operating system. In most cases, application level firewalls receive packet information directly from the operating system. This places an assumption that the underlying OS is reliable and secure. Unfortunately, this is not necessarily true. If an attacker is able to penetrate the operating system, an application level firewall can be rendered ineffective. It is for this reason that some commercial products have created secure, customized operating systems to support their application level firewalls.

2.2.2 Providing Public Services

In a protected network environment, there are often services to which "un-trusted" users should be or need to be given access. Examples include web (http), FTP, and DNS services. The most popular implementation for this scenario is to provide a separate network behind the firewall that implements a security policy, different from that of the primary network, that allows public access to certain services. This separated network is often called the "demilitarized zone," or DMZ (see Figure 2.6 on the following page).

A popular addition to a DMZ is to implement network address translation for the DMZ services. Each server on the DMZ can be assigned an external IP address for the firewall. Incoming requests for these services can then be translated by the firewall to their actual IP address on the network behind the firewall. Therefore, the DMZ configuration is able to provide separated public services while still maintaining a certain level of security.

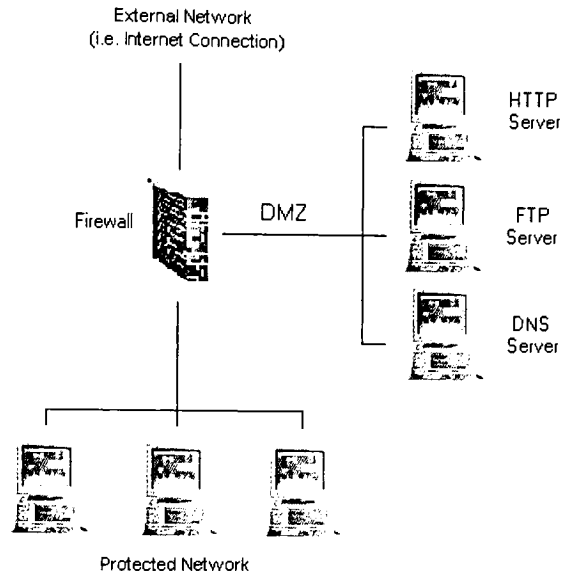


Figure 2.6 – Protected Network with a DMZ for Public Services

2.3 Existing Technology

As described in previous sections, there are a variety of firewall classifications, each of which adds a different aspect to firewall operation. Nearly every modern, high-end firewall application is constructed by combining various attributes from each of the basic types of firewalls into a single, more complicated solution. This section provides a brief overview of some existing firewall technologies, from [7].

2.3.1 Commercial Products

The most secure of modern firewall products are based on the stateful inspection architectures. Stateful Inspection was patented by Check Point Software in 1999 in their Firewall-1 product. A key of aspect of Firewall-1 is that it operates within the operating system, and begins inspecting packet information before the network layer in the protocol stack. This feature is similar to the kernel proxy strategy of performing security functions early in the protocol stack, in order to avoid accessing the application layer as little as possible. This, of course, improves performance. CYCON offers a second firewall solution based on Stateful Inspection called Labyrinth. In addition to the properties of stateful inspection, Labyrinth advertises network address translation (NAT) abilities along with "intelligent connection

tracking" (ICT). While NAT is a concept associated with circuit-level firewalls, the actual purpose of ICT is to perform the function of a circuit-level firewall. When a packet arrives, ICT circuit-level firewalls extract the IP and TCP connection information, which is then validated against a state table of existing connections. Of course, as part of Stateful Inspection, Labyrinth can also be configured to perform user authentication and dynamic filtering. Another Stateful Inspection firewall product is NetGuard's Guardian Firewall System. Guardian's architecture is unique in that its security operations occur beginning at the data link (i.e. MAC) layer. This is lower in the protocol stack than both Firewall-1 and Labyrinth. Sun Microsystems added their Sunscreen SPF (an Internet gateway firewall), EFS (encryption server with strong gateway functionality), and SKIP (end user and remote authenticated communication) models to the list of Stateful Inspection based firewalls. The Sunscreen line aims to provide an end-to-end suite of incrementally more secure firewall enforcement points.

Several modern architectures use the concepts of circuit level and dynamic packet filtering firewalls and in some cases have integrated these components into the OS kernel. CyberGuard's Firewall uses an approach that is called "dynamic stateful rule" technology, a version of dynamic packet filtering. Like dynamic packet filtering, CyberGuard's system provides packet-filtering functionality with the ability to create dynamic rules on a per connection basis. In addition, secure network and OS-level interfaces have been built directly into the operating system, thus providing a trusted foundation for the firewall application. A similar product is Milkyway's SecurIT FIREWALL. Like CyberGuard's Firewall, this product is based on an operating system that has been "hardened" to protect against attacks on insecure processes running at the OS level. The GNAT Box Firewall, from Global Technolgoy Associates, Inc. (GTA), is based on network address translation (NAT). As described before, NAT completely hides the internal network from the un-trusted world by eliminating the possibility of a direct connection between internal and external hosts.

Simple application gateway architectures (proxies) are also a popular basis for inexpensive firewalls. Trusted Information Systems' Gauntlet firewalls and Axent's Raptor Firewall are examples of such, providing sets of highly secure proxy services. These proxy services are able to perform the expected activities of application level firewalls, including user authentication and application command filtering. AltaVista provides another application gateway based firewall called the Active Firewall. As the name implies, a key feature of the firewall is that it attempts to

actively detect network attacks and security violations, and then reacts by alerting an administrator or shutting down the firewall. The InterLock Firewall from Advanced Network and Services, Inc. (ANS) and the Cyberwall Plus firewall from Network-1 are application level firewalls that attempt to perform intrusion detection (this is described in 2.5.3), in addition to application level filtering services. BorderWare from Secure Computing, IBM's eNetwork Firewall, and Ukiah's NetRoad FireWALL provide application gateway services that claim to enhance the firewall by using "multiple levels of security." These multiple levels of security are actually packet and circuit level filtering services that are provided in addition to the proxy services of the firewall. Like CyberGuard's firewall and the SecurIT firewall, BorderWare and IBM's eNetwork firewall are executed on security-hardened operating systems.

Secure Computing's Sidewinder Firewall provides a unique approach to an application gateway through its Type Enforcement technology. This firewall uses a specialized BSD Unix kernel that executes various applications in different "process domains." Presumably the processes will need to communicate across domain boundaries. Type Enforcement stipulates that any communication between any process domain and any data type must be given explicit permission. For example, an FTP session running in one process domain must have explicit permission to access a file in the data domain.

The following systems are comprised of a firewall appliance along with remote management software and are marketed as hardware/software systems. WatchGuard Technologies claims to have produced the first commercial firewall appliance called the WatchGuard System. This product, based on a Pentium system with remote management software, aims to provide a cost-effective firewall solution with all the major techniques for firewall implementation including packet filtering, proxies, and Stateful Inspection. Technologic's Interceptor Firewall is a hardware and software solution based on proxy services. Cisco's PIX Firewall appliance offers packet filtering and NAT services. It should be noted that most of these "hardware" systems are simply PC-based systems enclosed in a special, vendor marked case. A variety of routers with simple packet filtering capabilities also exist.

A list of firewall vendors is provided in Appendix A – Firewall Vendors.

2.3.2 Observations

Clearly the list of commercial firewall implementation is extensive. However, by studying these technologies, it is possible to make several observations about existing technology. To begin, the majority of firewalls are implemented as a software application. Many of these run on top of the existing operating system at the application level, and in some cases, security enhancements have been made to the operating system in order to improve performance. However, even when implemented at the OS level, the bottleneck problem is a reality for these (software-based) firewall systems. This is true because software cannot possibly run at the same speed as the native hardware being used in the network. Features such as proxy services and packet filtering will degrade the transparency of the firewall by either adding required steps for a user to access the protected network or by simply slowing down the apparent speed of the protected network.

In order to accommodate this fact, a trade-off must be made between firewall performance and the level of security provided. In most cases, the needs of the network determine which characteristic is more important for a given network. Therefore, solutions such as the ones mentioned previously have been developed to accommodate both cases. Networks requiring high performance and limited security are more fitting to firewalls performing basic packet filtering and possibly simple proxies services or NAT. Highly sensitive networks are better suited to firewalls performing all of the basic functions in addition to more complex operations such as Stateful Inspection and packet modification. In general, a firewall may be relatively fast and transparent, or it may be highly secure, but not both.

Another observation about modern firewalls is the aspect of programmability. Firewalls are highly configurable, but most are not programmable. It is generally not possible for an administrator to directly modify the code for a firewall. In some ways this makes sense. Making modifications to firewall code, especially at the kernel level, could expose holes in the security framework. However, a lack of programmability makes a firewall highly dependent on the existing technology, making it highly difficult to add new protocols, new applications, or company-specific traffic to the secure network. Few modern firewall implementations allow the ability to modify the firewall in this way.

The Stateful Inspection architecture, used in a number of the more complex firewall architectures, adds a high level of security to firewall architecture; however, the process of

inspecting the contents of a packet at all levels of the protocol (i.e. to perform user authentication and connection status monitoring) consumes a great deal of processing time involving memory management and application level processing. Because of these complexities, Stateful Inspection is obviously not suited for basic packet filtering and circuit level firewalls. It is for this reason that the latter, more primitive types of firewalls exhibit a lower level of security. Adding basic state monitoring to packet and circuit level firewalls would improve their security characteristics, while also maintaining high performance and transparency.

These comments about the abilities and shortcomings of existing technology are made in order to demonstrate areas of improvements for firewalls. It will be shown in Section 2.4.2 that network processors possess unique abilities that will allow such improvements.

2.4 Importance of Network Processors

The network processor is a very recent addition to the arena of networking hardware. This section describes the forces behind the advent of the network processor, some of their hardware characteristics and abilities, and their relevance to improving firewall technology.

2.4.1 Emergence of Network Processors

As mentioned several times before, there has been an expansive growth in networking technology over the last few years. One of the main forces behind this is the nearly exponential growth of the Internet. High demands on network backbones, especially due to the increased use of voice and video over the Internet, have made network congestion a serious problem. Lack of bandwidth is often blamed for such congestion, and the addition of higher-bandwidth network hardware, such as fiber optics, is often used to solve the problem [12]. However, the process of upgrading existing hardware can be an expensive process; therefore, organizations are now looking for ways to improve the efficiency of their networks, rather than constantly upgrading systems.

These efficiency problems are usually bottlenecks caused by application servers or routers, hardware that performs network services such as load balancing, quality of service (QoS) functionality, gateways, and security (i.e. firewalls). Network processors were developed in order to improve or to even eliminate these bottlenecks through more efficient utilization of existing network resources and by providing network services at wire speeds. By offering these

performance improvements, manufacturers hope that network processors will emerge as the "building blocks" of networks in the future.

In [12], a network processor is defined as a programmable communications integrated circuit capable of performing one or more of the following functions: packet classification, packet modification, queue/policy management, and/or packet forwarding. Packet classification involves identifying a packet based on a known protocol. Packet modification, as expected, means making direct changes to a packet's content. An example would be modifying header fields such as source/destination address or port number in IP or TCP packets respectively. Queue/policy management deals with the scheduling of packets for specific applications with respect to the particular design strategy for packet queuing and de-queuing. Lastly, packet forwarding is the transmission and receipt of data over a switch fabric and the forwarding of the packet to the appropriate address. It should be noted that these are just the basic characteristics of network processors. While they are capable of performing these functions, their intended power involves the combination of these abilities into higher-level applications, such as for network security.

As network processors are a growing technology in the communications and Internet communities, manufacturers are no less than racing to establish a market share. A list of network processor vendors is provided in Appendix B – Network Processor Vendors. A majority of these companies are participating in the Network Processor Conference scheduled for October 23-25, 2001 (NPC 2001) in San Jose, California.

2.4.2 Relevance of Network Processors to Firewall Technology

As described in previous sections, the evolution of firewall technology has made firewalls a formidable component of network security. However, there is still a sharp trade-off between network security and performance. Simple packet filter and circuit-level firewalls provide high speed network traffic filtering, but are completely vulnerable to content related attacks. Just the opposite is true in the case of more complex firewalls that implement stateful inspection or kernel level proxying. The high level of security provided by the latter type of systems can cause bottlenecks in the protected network. These bottlenecks occur because the operations necessary to ensure a high degree of security require must typically be implemented in software. Network processors possess a number of characteristics that hold promise for improving the

performance of existing firewall technologies and also lessening the existing gap between performance and level of security.

One example is the ability to manipulate packet header information, such as IP source and destination addresses or transport layer port numbers. This could be used to perform operations like network address translation at high speeds and would eliminate the need to reconstruct or copy packets in memory before transmitting the modified packets. Allowing the firewall to modify the packets in this way would improve the speed and potentially the transparency of the firewall. In addition, many current firewalls that execute in the application domain (i.e. proxy services) do not have direct access to information contained in low-level packets. Implementing higher-level firewall concepts (such as proxies) on a network processor would allow direct packet modification at any point in the protocol stack.

Another important characteristic of network processors is the ability to interpret application level protocols. This ability typically exists in higher-level firewalls, those that are based on application level services. Traditionally, giving low-level filters knowledge of application level information would deteriorate the performance of the firewall (its main benefit, in the case of low-level firewalls). In the case of network processors, the ability to directly observe application layer packet content combined with wire speed processing rates indicates the potential to perform application level filtering while maintaining the performance of simple low-level firewalls. These abilities could be expanded further to include services such as user authentication and virus detection.

Using the memory capabilities of network processor platforms, it is conceivable that network processors will also be able to maintain state information on connections that have occurred across the firewall. This ability could allow network processors to implement concepts used in stateful inspection firewalls. More importantly, the high speed processing abilities of network processors will allow these operations to be performed at rates much higher than traditional stateful inspection systems. As stateful inspection relies on knowledge of application level information, this potential is also an extension of deep packet processing abilities.

As shown in the previous section, network processors are being developed by a number of networking manufacturers, each of which provides unique architectural characteristics for their product. These unique qualities could also provide the ability for firewall performance improvement. A significant example is an architectural pattern of embedding a certain number

of packet processors along with special purpose co-processors within the network processor chip. The presence of multiple processing elements means the ability to explore parallel processing algorithms. For example, the ability to apply filter rules to a number of packets in parallel or the ability to pipeline protocol comparisons would significantly speed-up the firewall.

In summary, network processors allow the standard execution of filtering rules to occur at hardware speeds, but also offer packet processing and modification capabilities at speeds typically not available in current firewall solutions. This suggests that network processors are potential replacements for existing hardware-based firewall implementations such as filtering routers and application level firewalls. This thesis explores the practicality of implementing firewalls using the capabilities provided by an example network processor, the IBM PowerNP 4GS3.

2.5 Current Related Research

While firewalls continue to be a fundamental element in providing secure networks, there is surprisingly little research in the area of firewall technology. This is because firewalls have advanced to the point where they are considered one of the basic building blocks of a network. Rather than focusing on direct improvements to the firewall, researchers are targeting larger scale topics of network security, using the firewall as a tool for implementation. Several example topics are described in the following sections, along with implementation possibilities for network processors

2.5.1 Virus Detection

Virus threats are a problem facing the modern networked community. They can range from simple annoyances to infections causing devastating system failures. Approaches to virus detection have traditionally been based on pattern recognition. Packets or files are scanned for byte patterns of known viruses, or in more recent systems, are searched for certain characteristic sub-strings. In both cases, the virus detection algorithm is based on static characteristics of the virus [24]. This property makes pattern recognition a purely reactionary approach. In other words, the virus detection software reacts based on previous knowledge, and must be manually updated to detect a new virus string. Of course, manual updates cannot occur until after the

new virus has been introduced, usually on the order of months, most likely after damage has already been caused. Pattern recognition systems are simply not effective against newly introduced viruses.

Unfortunately, new viruses will continue to be developed and deployed, but with continuingly more complex functionality. For this reason, research is being devoted towards the creation of pro-active systems that will be able to algorithmically detect and take steps towards eliminating the virus without direct human intervention. These algorithms are, in most cases, based on heuristics. However, it is difficult to know how such algorithms will work or what problems will be caused until they are tested in a real situation. For this reason, it is necessary for researchers to understand new algorithms from an analytical perspective before anti-virus solutions are deployed [24].

After a heuristic for virus detection has been developed and simulated, implementation will become an issue. The modern scenario for virus propagation involves the virus spreading itself via a large network, such as the Internet. This is often done through mail applications; however, it is now a greater reality because of the rapid emergence of handheld, mobile data devices, where anti-virus software has not been completely employed as it has been in the PC industry. This propagation of viruses over large networks, potentially aided by the existence of mobile devices, has led to distributed approaches to virus detection. Research in virus detection is being led by a variety of organizations including anti-virus software manufacturers such as Symantec, McAfee, and Norton.

Because the firewall is focal point for network traffic, it makes sense that the firewall machine would be a likely point to implement virus detection. In many cases, the virus is contained at the application level of packet content, and it is already a feature of many modern firewalls to perform inspection on application layer content. Thus, the firewall is a likely candidate for the implementation of, or at least a starting point for, virus detection. It should be remembered, however, that the efficiency of the firewall is a key issue. Algorithmic approaches to virus detection would undoubtedly affect the speed of the firewall. In addition, the firewall system would need to have significant processing power in order to perform algorithmic detection applications. Filtering hardware systems, built entirely for speed would typically not have this power, and the additional computation added to an application level firewall may cause a bottleneck.

Network processors possess the ability to perform packet inspection, but also have the ability to perform such inspections at wire speed. It is typically possible to communicate directly with the network processor (i.e. network processors have the ability to communicate with external machines), thus distributed applications could be feasibly implemented using network processors. Therefore, by combining high rate computational abilities and wire speeds for packet processing, network processors appear to be a reasonable implementation platform for implementing virus detection as part of a firewall system.

2.5.2 Virtual Private Networks (VPNs)

Another related area of research is virtual private networks (VPNs). A definition provided in [10] defines a VPN as a network using encryption and tunneling techniques to connect users or sites over a public network, typically the Internet. An example of this type of network is shown below.

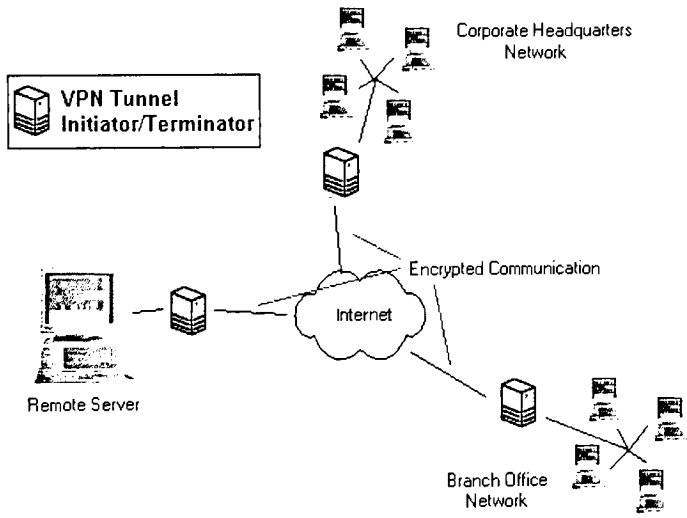


Figure 2.7 – Sample Virtual Private Network

In other words, a VPN is comprised of a physically disjoint networks or individual machines separated by an un-trusted network (i.e. the Internet). When information from one site is destined for another, the information is encrypted, transmitted across the un-trusted network, and decrypted at the target. This technique is known as tunneling.

Like a firewall, the tunnel initiator/terminator must be the only access point to the protected network. Because of this, VPN functionality is now being integrated into many modern firewall

products. Another similarity to firewalls is that a high dependency exists on the security of the VPN server. If it is possible to breach the security mechanisms of the server, the cryptographic encapsulation of transmitting data is relatively useless, as transmissions could still be intercepted.

Adding the encryption functionality used by VPNs into the security operations of the firewall further strengthens the security of the protected network. However, as with virus detection, this increase of complexity in the firewall system can lead to a bottleneck in the network. Network processors are once again a promising solution to implementing VPNs within the context of a firewall. The implications of network processors to firewalls have already been described. Many of their architectural features, especially wire speed packet processing capabilities and programmability, lend themselves naturally to the implementation of VPN encryption within the context of a firewall.

2.5.3 Intrusion Detection

One conclusion that can be reached from the modern networking environment is that current security measures, including firewalls, do not guarantee absolute security. For this reason, additional security measures are being pursued. Intrusion detection has become a popular field aimed at providing security above and beyond the firewall.

In [2], intrusion detection is defined as the process of monitoring the events occurring in a computer network, analyzing them for signs of security problems. The basic concepts involved can be extracted from this definition. In order to analyze the "events" in a network, there must first exist a system that produces a set of data that describes what is occurring in the network, a "stream of event records" according to [2]. Next, there must be a portion of the system that analyzes the data. Finally, a component of the system must react to the data based on the security policy of the protected network. While this is a rather straightforward description, implementation is far from trivial (and beyond the scope of this document).

The relevance of firewalls to intrusion detection relates back to the first key aspect of an intrusion detection system: a source of network event data. As described in 2.1.2, an important characteristic of a firewall is to perform logging of the network traffic across the firewall. The firewall log can be a significant source of the data needed by an intrusion detection system. However, as in the case of the firewall, adding intrusion detection as a security measure for the network can potentially affect the performance of the network. The programmability and wire

speed packet processing abilities of network processors once again seem applicable. A network processor based firewall would be a high-speed alternative to generating network traffic logs for an intrusion detection system. In addition to VPNs and virus detection, network processor based firewalls provide important performance implications to intrusion detection.

2.6 Chapter Summary

This chapter has provided a broad background including the history, present state, and future of firewalls in the field of network security. As firewalls continue to be a building block for the secure network, it is important to continue the improvement of firewall technology. It has been shown that the key issue in firewall implementation is the tradeoff between level of security and the performance of the network protected by the firewall. Network processors, because of their unique abilities to perform high-speed packet processing and packet manipulation, have the potential to lessen this gap between performance and level of security. The following chapter introduces several strategies for the implementation of firewall systems on a network processor, using the IBM 4GS3 Network Processor as an example platform.

Chapter 3

Theory

3.1 Proposed Firewall Improvements

From the information described in the previous chapters, it is clear that there are numerous characteristics associated with a network processor's architecture that in theory could be used to enhance the performance of a firewall. This section proposes a set of improvements to firewalls that are made possible by network processors, using the abilities possessed by the IBM PowerNP 4GS3 (hereafter referred to as 'NP') as a target architecture.

The first, and foremost, improvement that can be made is in performance. Because network processors have the ability to process packets at wire speeds, it is logical that simply implementing a firewall on a network processor, a processor designed specifically for packet analysis and manipulation, will provide better performance than existing filtering routers or similar technology based on general purpose processors. For this reason, the first portion of this study is to create a simple packet filter, the most basic of firewall architectures (see section 3.2).

A second characteristic of network processors that will improve firewall performance is parallelism available within the processor. Within the core of the NP are multiple instances of smaller packet processors. When a packet arrives from the network, it is dispatched to one of the currently idle processors. Therefore, the NP is capable of processing multiple packets in parallel. In addition, the NP is a multi-threaded architecture, which further reduces stall cycles that might occur while satisfying instruction dependencies.

An even more significant opportunity for parallelism is provided by the co-processors that are part of the NP core. For example, in the IBM PowerNP 4GS3 memory accesses are performed by a specific co-processor. When a memory operation is initiated, the packet processor may continue performing operations on the packet while allowing the memory access to complete. In the case of packet filtering, several memory look-ups must be done for each

packet that enters the firewall. By overlaying the firewall operations with memory look-ups the speed of the firewall will improve. These concepts are outlined more specifically in the section describing the basic NP packet filter (section 3.2).

For a final improvement to firewall performance, a parallel firewall architecture could be created using a combination of multiple network processors. A theoretical model of a parallel firewall architecture, obtained from [3], is shown below.

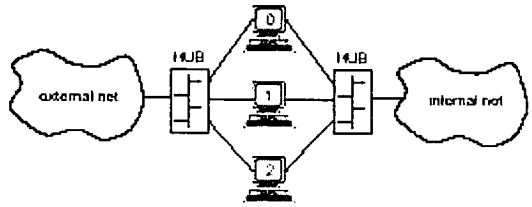


Figure 3.1 – Parallel Firewall Model

In the parallel firewall model, packets traveling between the internal network and the external network view the parallel firewall as a single entity (i.e. one line in, one line out). However, the work of packet filtering is shared between the individual machines within the parallel firewall. While figure 3.1 shows the firewall machines as individual workstations, these could be implemented using network processors. A second benefit of this architecture is that it provides redundancy to the firewall in the event that one firewall machines crashes or goes offline. If a crash is detected, the remaining firewall machines could be instructed to perform the packet filtering typically done by the failed machine. In this configuration, the network processors share the load of packet filtering with the purpose of lowering the possibility of a bottleneck caused by the firewall while also providing a degree of transparency in the event that one of the firewall machines is removed from the network. An approach to implementing a parallel firewall using network processors is described further in section 3.3.

3.2 Basic NP Packet Filter

The method of packet filtering is a traditional approach to firewall implementation. As discussed earlier, packets are permitted to pass through the firewall based on a set of properties that define the firewall's security policy, the firewall "rule." The diagram below shows the basic elements of a firewall system utilizing a network processor. As shown, the

firewall is composed of a network processor that performs the filtering operations, and a control point system that allows user management of the network processor. An important requirement is that the firewall must be the single point of access to the protected network.

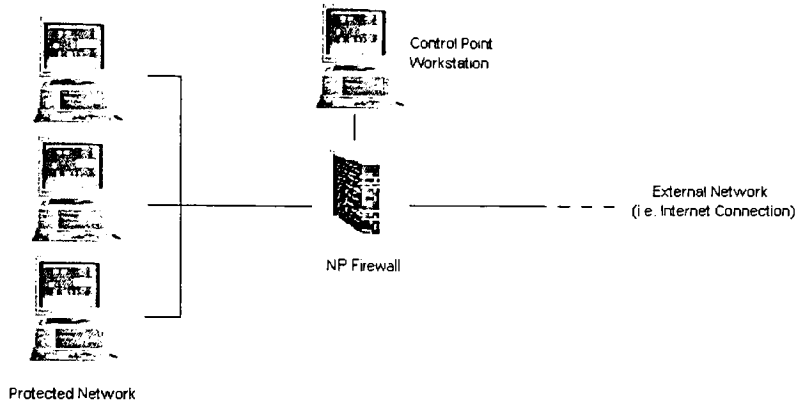


Figure 3.2 – Basic NP Firewall

There exist two basic approaches to developing a firewall rule. The firewall can either a.) permit all network traffic unless it is explicitly denied by the rule or b.) deny all traffic unless it is explicitly permitted by the rule. It should be reminded that the first option is inherently less safe than the second, as the default action is to permit traffic to pass through the firewall. This clearly increases the set of potential security violations.

The firewall rule can be based upon a number of properties pertaining to the network traffic. It is important to first verify the protocols being used by network traffic. This becomes the first condition when defining a firewall rule. In the Internet, the protocols of choice are IP for routing and TCP for transport layer functionality. The firewalls designed for this project are based on this premise. The relevant properties used for defining the firewall rule are the IP source and destination address and the TCP source and destination port numbers. This information is readily available from the IP and TCP headers.

When designing the firewall for the network processor, several issues must be addressed. A method must be designed for modeling a firewall rule. Then, the complete rule will be composed of a set of individual rules for defining specific IP and TCP traffic that is permitted to pass through the firewall. The complete rule can be conceptualized as a table of the individual rules. Because the latency caused by the firewall application must be minimized, the latency in

traversing the rule table should also be minimized. This becomes especially true when considering that multiple look-ups must be performed for each packet (for the IP source address, IP destination address, TCP source port, and TCP destination port). If a traditional list or array structure was used, the latency in performing a rule would be O(N). A more desirable structure for maintaining the firewall rules would be a tree structure, one of O(log N).

The NP being used for this project, in fact, possesses a memory architecture that is organized around maintaining data structures as trees. For every element that is placed in the tree, an associated key must be given. When attempting to look-up a data element, a search key is generated and then used by the branch nodes to perform comparisons with the keys associated with the leaf data.

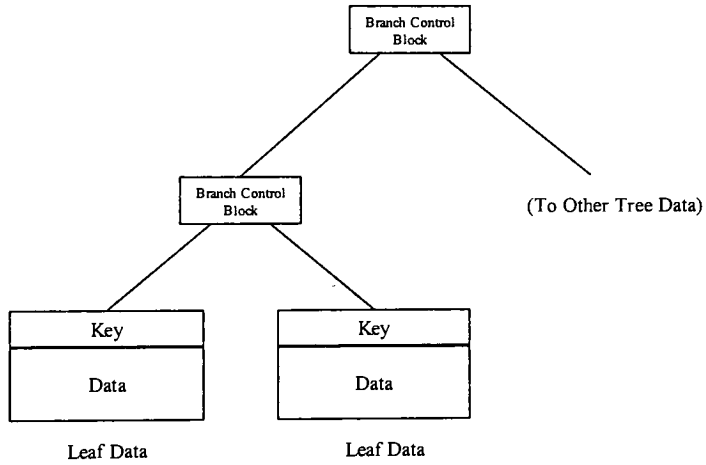
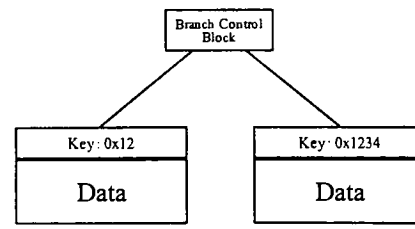


Figure 3.3 – Generic NP Tree Structure

This brings up the concept of two types of search algorithms provided by the NP hardware to traverse the tree: Longest Prefix Match (LPM) and FPM (Full Pattern Match). In an LPM search, the leaf key with longest sequence of matching bits is returned for a given search key.

LPM Example:
Two leaves with keys 0x12 and 0x1234 respectively

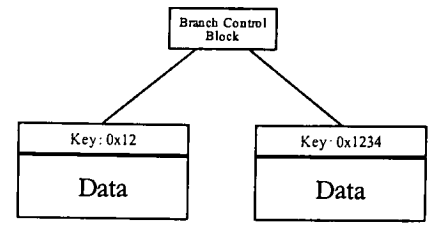
Suppose the following searches:
Search Key: 0x1299 returns leaf with key 0x12
Search Key: 0x1234 returns leaf with key 0x1234
Search Key: 0x1 returns no leaf



The FPM search has a more obvious behavior – a leaf is returned by the search when the search key exact matches the bits in the leaf key (stated alternatively, the bit pattern of the leaf key fully matched the bit pattern of the search key).

FPM Example:
Two leaves with keys 0x12 and 0x1234 respectively

Suppose the following searches:
Search Key: 0x1299 returns no leaf
Search Key: 0x1234 returns leaf with key 0x1234
Search Key: 0x1 returns no leaf



These two types of search algorithms relate directly to the data searches that would be made when attempting to look up IP and TCP rules for firewalls. In IP filters, it is often desirable to filter packets destined to a particular network, for example anything on the 129.21.xx.xx. It would require approximately 2^{16} individual rules to filter packets destined for any IP address on that network, given a standard IPv4 32-bit network address. By filtering on just the upper 16 bits of the IP address, this filter can be express as a single rule. This is exactly the purpose of the LPM search. By matching a key based on the upper portion of a subnet address, all of the specific, full length IP address can be filtered. Filters on specific IP address on the given subnet may be removed by adding individual permissive rules thereafter.

TCP filters on the other hand are based on specific port numbers. An additional implication is that the packet is actually a TCP packet. In other words, the firewall rule is based not just on the port number, but also the protocol type. It is possible that other transport protocols, such as UDP, may be used on the network that share a TCP port number with another application. For this reason, the filter rule must match both the protocol and the port number. This characteristic of exact matching implies usage of an FPM search on the network processor. The specific details of the implication of the IP LPM rule tree and the transport FPM rule tree are discussed in detail in section 4.2.

Another important piece of information that must be mentioned is that tree searches on the NP are performed using special purpose co-processors. After a search key has been generated, the tree search (i.e. memory search) may be issued to the co-processor allowing the packet

processor to continue processing of either the packet information, or preparation for the next firewall rule look up (again, multiple look-ups are needed for each packet). Below is a flowchart outlining an algorithm for performing the firewall rule look-up that demonstrates usage of parallelism in the NP.

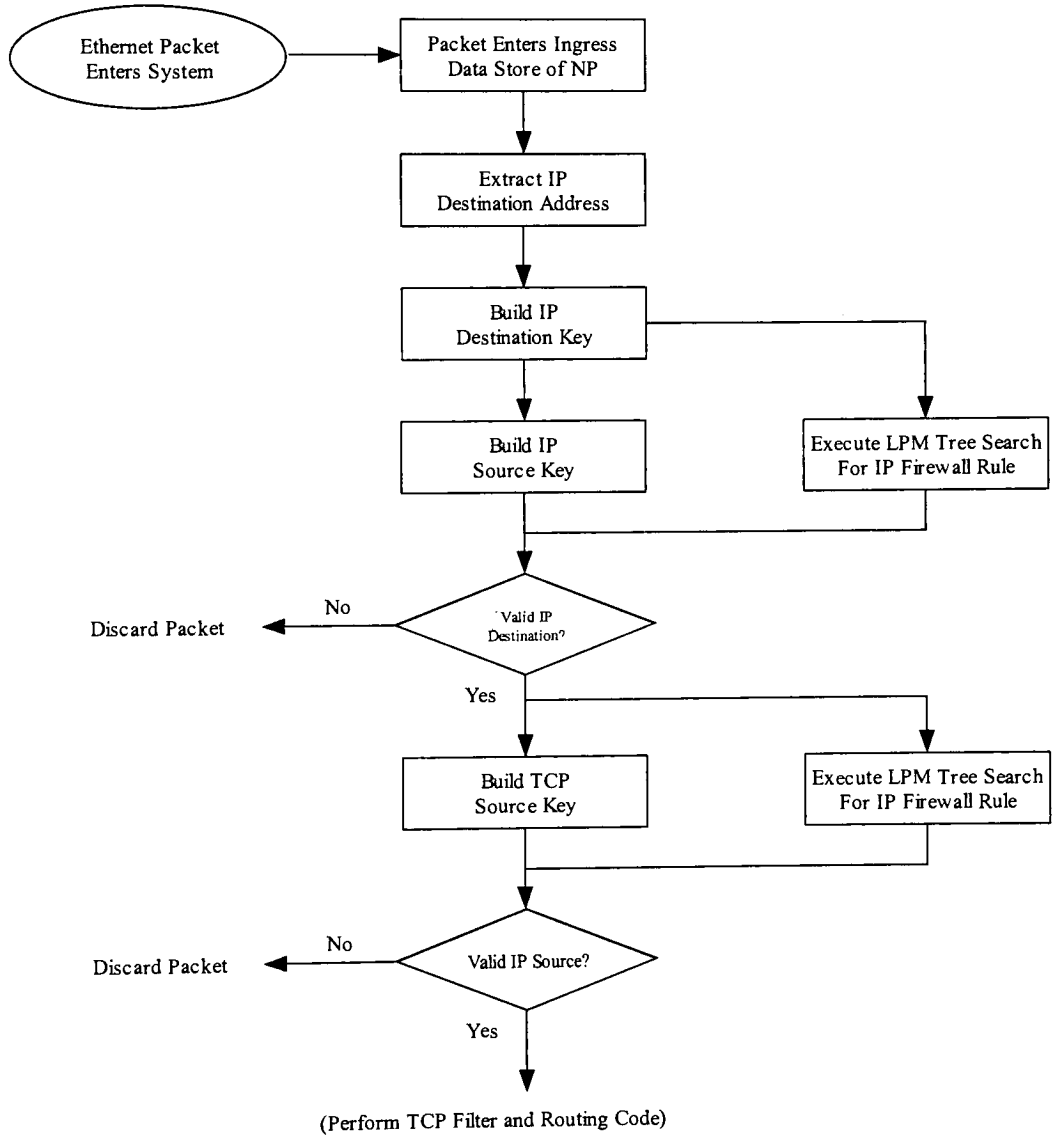


Figure 3.4 – Packet Filter Algorithm (IP Portion)

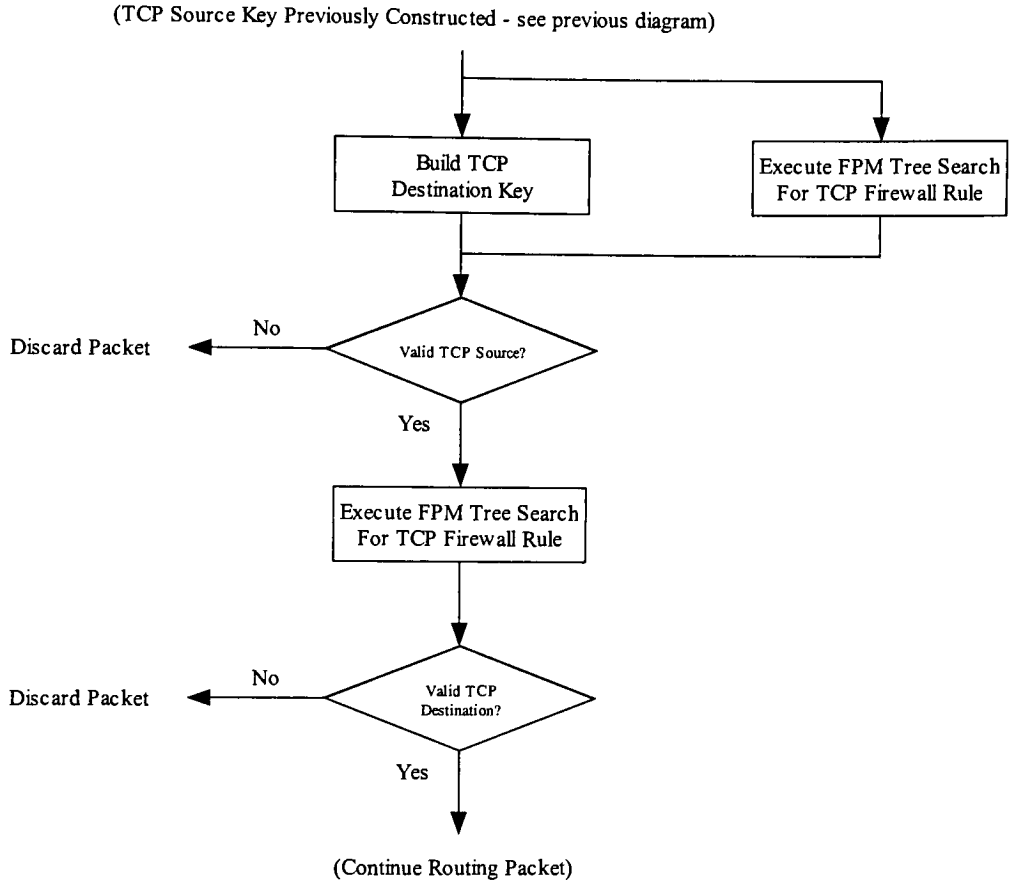


Figure 3.5 – Packet Filter Algorithm (TCP Portion)

This algorithm uses parallelism by overlapping the creation of search trees keys with the look up of firewall rules. Utilizing the tree method of data storage provided by the NP along with the parallelism that is made available during tree look-ups by the memory co-processor suggests significant performance benefits to the firewall.

It is also necessary to determine a data format for the representation of a rule element in the table. Next, the network processor provides a variety of memory options. Memory usage (i.e. the type of memory – DRAM, SRAM, etc) and associated memory properties will also affect the latency of the firewall. Lastly, the rule table should be managed in such a way that rules may be added or deleted in a simple fashion. All of these issues must be addressed; however, they are more appropriately discussed in terms of specific implementation details, as described in section 4.1.

3.3 Parallel Packet Filter

One of the key drawbacks to using a firewall is the potential for causing a bottleneck in the network. Consider the basic packet filtering algorithm described in the previous section. In order to perform the most basic of firewall activities, four memory accesses are necessary. Memory accesses are extremely costly to the performance of a high speed processor, and henceforth to the firewall. Therefore, attempting to distribute the load of the filtering operations, as depicted by the parallel firewall model in figure 3.1, can potentially lessen the potential for causing a bottleneck.

While the parallel firewall model at first appears to be a simple architecture, there are a number of realistic complications that occur in its design. Most of these complications are due to constraints caused by standard protocols used by the Internet. To begin, in order for a firewall to transmit packets from one side of the network to the other, it must have some notion of routing. It can be assumed that the packet filter being conceptualized for the firewall is also a router, or an example of the so-called filtering routers that have been mentioned previously in this document. It is typical for standard IP routers to have a designated IP address on both the internal and external side of the network. Hosts on the internal network are configured with the address of the router on their network so that packets can be routed to the external network (i.e. the Internet). The same would be true for a firewall. When a packet on the network must be routed to the external side of the firewall/router, the firewall/router must communicate with the host transmitting the packet. In order to do this, it is necessary for the router and the host to communicate at Layer 2 of the network protocol stack (the Datalink layer – used to provide error correction during data transmission.). One of the most common Layer 2 standards is Ethernet, which makes use of the Address Resolution Protocol (ARP). This protocol involves a direct mapping between the hardware-assigned ARP address of a network card and its configured IP address. As stated previously, the router interface (on either side of the network) is configured with an IP address that allows the router to respond to ARP requests from one of the network hosts. The parallel firewall diagram shown in Figure 3.1, however, violates this single interface model to some degree because the firewall/router has three connections to the network (via a hub), which implies the need for three IP addresses and therefore three ARP addresses. The fundamental problem that thus occurs is how to resolve the problem of network

hosts attempting to send data to, conceptually, a single firewall/router interface, and having multiple router/firewall interfaces arbitrate the request for data transmission over ARP.

There are several solutions to this problem. One solution is to program the hosts to alternate the address to which they send outgoing packets. This of course means adding a workload to the host machine, and therefore a reduction in the apparent performance of the network. More importantly, this would require a modification to the operating system of each host on the network. These two drawbacks clearly make this solution a poor alternative for implementing a parallel firewall.

A second solution is to configure each of the firewall/router machines with the same network address and synchronize the parallel machines "taking turns" in responding to a packet transmission and performing the subsequent filtering rules. In order to do this, a means of communication is required between the parallel machines is required. This synchronization; however, would be a relatively complicated process and would lower the performance of the firewall. This would be especially true if the inter-machine communication occurred on the target networks at a level that would cause additional network contention. A second obstacle to this architecture is that direct modification to the system executing the Layer 2 communication might be necessary. This processing is often done by the network interface card hardware, making it difficult, if not impossible, to modify. In reality, a network processor would be capable of coordinating this type of architecture, because Layer 2 protocols may be implemented on a network processor, and therefore be modified at the assembly code level. However, given the difficulty of coordinating the inter-machine synchronization, the following architecture is more practical in terms of implementation.

A final architecture, the one implemented for the purposes of this thesis, follows the model of a single interface to the firewall/router. This is done by adding an additional machine that acts as the single interface to the firewall/router. This machine is then responsible for performing a simple switching operation between the parallel firewall machines (see figure 3.6 on the following page).

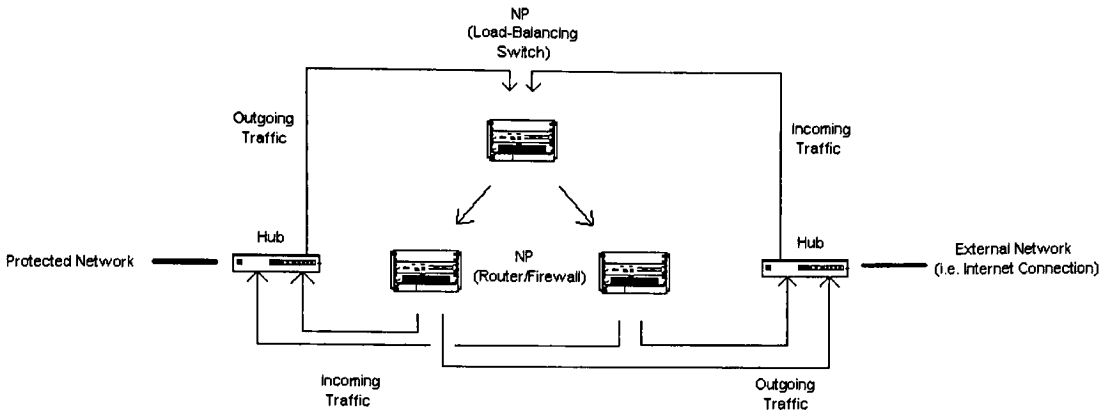


Figure 3.6 – Switch-based Parallel Firewall

In this architecture, network hosts are configured to send traffic to the switch NP. Packets passing through the switch are directed to one of the firewall/router machines, which then performs the more time-consuming packet filtering operations. A simple algorithm would be to alternate between the two firewall/router machines. As can be seen, the switch adds a second hop to the firewall/router. Therefore, the speed of the switching machine must be fast enough to not outweigh the performance benefit achieved by sharing the load of packet filtering provided by the parallelism of the firewall/router machines. A network processor is capable of performing this type of switching operation at extremely high speeds.

This architecture possesses two additional characteristics that are worthy of mention. First, the synchronization that would be required in the previous architecture is no longer necessary because the arbitration of packet processing is centralized in the switching machine. This also avoids amount of network added by the parallel firewall system. Secondly, this architecture is capable of providing the redundancy mentioned in section 3.1. In the event one of the firewall machines fails, the switch can begin directing network traffic to one of the remaining firewall/router machines. On the other hand, this firewall relies on the switch being online at all times. The specific implementation of this firewall is described in section 4.3.

3.4 Chapter Summary

Network processors provide a wide range of functionality that can be used in firewall applications. While they can be used to implement simple packet-filters, network processors can also be used as building blocks for complex firewall architectures, such as the parallel firewall. The next chapter describes the specific implementation details of a basic NP packet filter and a parallel firewall system using the architectures described in the previous two sections. These implementations illustrate the usage of the high-speed packet processing abilities and advanced architectural features, including tree-based memory structures and parallelism provided by co-processors, provided by the IBM PowerNP 4GS3.

Chapter 4

Implementation

4.1 Development Environment

Before discussing the details of the two NP firewalls that were developed, a brief description of the development environment should be made. To begin, applications for the IBM PowerNP 4GS3 network processor (referred to hereafter as "NP") are developed in two parts: the actual NP assembly code, termed "picocode," and the application management control software, termed the control-point. The diagram below depicts an NP system with an external control point machine. In reality, the control point could be implemented on a different platform, such as an embedded PowerPC core residing on the NP die.

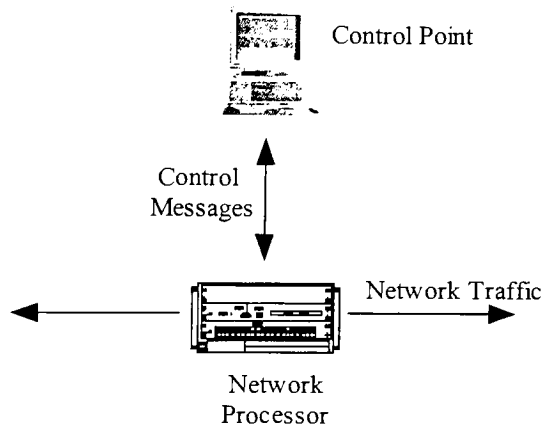


Figure 4.1 – NP System Diagram

The firewall applications were created using IBM's Advance Software Offering (ASO) for the PowerNP 4GS3. This package, targeted for Linux V6.1, includes a suite of code development tools in addition to a router application implemented on a simulated NP. The following are primary components included in the package:

1. *NP Assembler* (npasm)– Compiler for PowerNP 4GS3 picocode.
2. *NP Scope* (npscope) – Debugger application used to test picococde and test memory structures, such as the tree structures discussed in the previous chapter.
3. *NP Simulator* (npsim)– Linux application that simulates a network processor blade with an attached 40-port Ethernet board.
4. *NP Router Picocode* – A complete set of boot and application picocode that gives the NP full dynamic routing capabilities when used in conjunction with an external control point system.
5. *NP Control Point* (npcp) A control point application, implemented in C, used to manage the NP router application.
6. *NP Control Point Console* (npcp console) – A simple text-based user interface to npcp.
7. *GateD Public Release 3.6 (see www.nexthop.com) - Linux application that manages OSPF routing tables, used in conjunction with the NP routing application*

The simulated router application provided by the ASO was used as the basis for the firewalls developed using the NP. In order for a firewall to transmit packets, it must have at least some basic level of routing capabilities. Therefore, the packet forwarding abilities provided by the router application served as a perfect starting point to begin adding firewall functionality. Thus, the approach taken for this project was to create a firewall by upgrading the abilities of the existing routing application to include packet-filtering capabilities.

4.2 Basic Packet Filter

As discussed in section 3.2, the purpose of the basic NP packet filter was to utilize the tree method of data storage provided by the NP along with the parallelism that is made available during tree look-ups by the memory co-processor. In order to do this, it was necessary to design a memory scheme for managing the IP and transport firewall rule tables and add code to the existing routing application picocode and control point code for the purpose of managing the firewall data structures.

4.2.1 NP Hardware Tree Management

While the NP does in fact manage tables as logical tree structures, the hardware representation is somewhat more complex. In the NP hardware, all data elements are maintained as leaves, and branching information is stored in node structures called PSCBs

(Pattern Search Control Blocks). The PSCBs are then accessed by a hashing table, (called DTs – Direct Tables) in order to improve efficiency. Each DT entry points to a unique PSCB. An example is shown below.

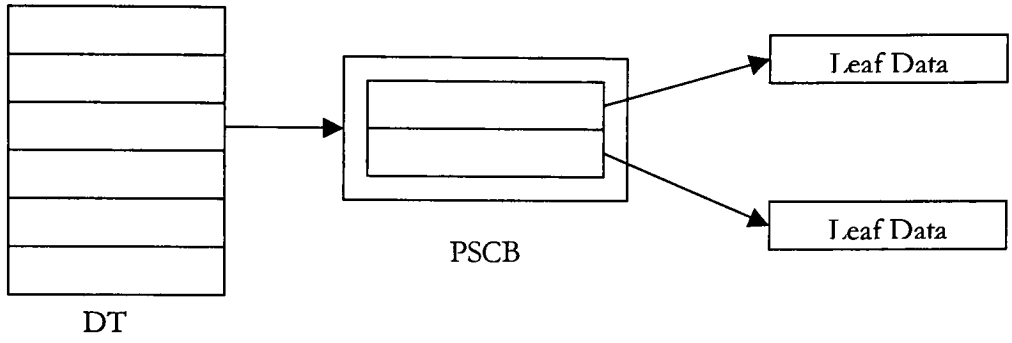


Figure 4.2 – NP Tree Structure Hardware Model

Thus, when defining a tree structure, it is necessary to allocate memory resources for the DTs, PSCBs, and leaves. There are a variety of different memory resources available, which range in terms of data width and speed. After declaration, DTs and PSCBs are fixed in size; however, the quantity and size of leaf structures will depend on the application executes. Obviously, memory must be allocated in order to store all of the necessary leaves. The specific memory allocation for the firewall rules is described in the following section.

4.2.2 IP Firewall Rule Tree Structures Design

Section 3.2 introduced the concepts relating to the tree managed data structures provided by the NP. It was decided to implement filtering in the basic NP firewall based on IP (Internet Protocol) source and destination address and transport layer source and destination port for and TCP (Transport Control Protocol) and UDP (User Datagram Protocol).

As mentioned previously, it is often desirable for an IP filter to filter packets at the network level rather than on specific IP address. This functionality is well suited for the Longest Prefix Match (LPM) tree structure provided by the NP. In order to implement an LPM tree for managing the IP firewall rules, it was necessary to create a memory scheme (a combination of key/leaf objects) for accessing the rule information contained in the trees at leaf nodes. Because Internet traffic is guaranteed to use the IP protocol, the key for accessing the firewall rule table

was based entirely on the IP address. For reference, an IPv4 (Internet Protocol Version 4) address is represented in the form AA.BB.CC.DD, where each of the four portions of the address may be a number from 0 – 255. This address is stored and transmitted as a 32 bit integer according to the equation:

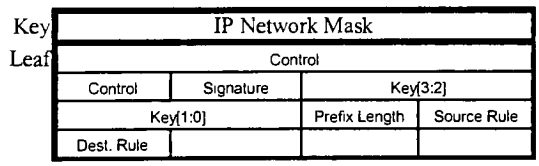
Equation 4.1 - Integer Value of IP Address = (AA << 24) | (BB << 16) | (CC << 8) | (DD)

The integer value can then be used directly as a key for the LPM search used to index into the IP rule table. Refer to figure 3.3 for a diagram of the tree structure. Below is an outline of the memory design process.

Important Notes:

An IP rule depends on either the IP source or destination field, the respective IP net mask, and the number of bits from the net mask that are used as part of the key (prefix length).
It is useful to work with the parameters on a byte by byte basis
It is a hardware requirement for the key to be a multiple of 16 bits – the IP address is 32 bits so this is not a problem
The data leaf includes all information including a copy of the data used to build the search key.
Control overhead for the leaves is 5 bytes

IP Rule Leaf Structure



***Key Bytes*:**
Field:
IP net: aa.bb.cc.dd - 4 Bytes

***Leaf Data*:**
Field:
Control *(Used by NP hardware to manage the order of elements in the tree)*: 5 Bytes
Signature *(Identifies the Control Point that created the rule)*: 1 Byte
IP net *(Copy of the key)*: 4 Bytes
Prefix Length *(Number of bits of the key used to perform the LPM look-up)*: 1 Byte
Source Rule *(Action to perform if the IP net is from the source field of a packet)*: 1 Byte
Destination Rule *(Action to perform if the IP net is from the destination field of a packet)*: 1 Byte

***Byte Codes for 'Source Rule' and 'Destination Rule' Fields*:**
Firewall Action:
Permit = 0x00
Deny = 0x01

***Summary*:**
Control Overhead - 5 Bytes
Key - 4 Bytes
Leaf Data - 8 Bytes
Total: 17 Bytes = 136 Bits

This detailed byte information is necessary for allocating memory blocks within the network processor. In the PowerNP 4GS3, memory allocation is highly dependent on the dimensions (width and height) of the memory block being used. For example, DRAM was chosen for the firewall rules because of the size of the module. The firewall rule list could potentially grow to a

large size; therefore, causing memory overflow. The NP possesses four DRAM modules each with four banks. Most of these modules are 64 bits wide. Each IP rule leaf structure, shown above, uses a total of 17 bytes or 136 bits, which implies three DRAM banks. Thus, the memory allocation for IP rule leaves was a width of three and a height of one. Several example leaf structures are shown below.

Example Rule #1:
"Deny any packet destined for the 129.21.26.139 address"

This implies a prefix length of 32 bits = 0x20 =>
Key: 0x81151A8B = (129 << 24) | (21 << 16) | (26 << 8) | (139) using Equation 4.1
Leaf Data, excluding hardware control bits (suppose Signature = 0x01): 0x0181151A8B200001

Example IP Rule Leaf Structure (see previous page)

Key	0x81151A8B			
Leaf	Control			
	Control	0x01	0x8115	
	0x1A8B		0x20	0x00
	0x01	un-used	un-used	un-used

Example Rule #2:
"Deny any packet originating from the 207.46.xx.xx network"

This implies a prefix length of 16 bits = 0x10 =>
Key: 0xCF2E0000 = (207 << 24) | (46 << 16) | (0 << 8) | (0) using Equation 4.1
Leaf Data, excluding hardware control bits (suppose Signature = 0x01): 0x01CF2E0000100100

Example IP Rule Leaf Structure (see previous page)

Key	0xCF2E0000			
Leaf	Control			
	Control	0x01	0xCF2E	
	0x0000		0x10	0x01
	0x00	un-used	un-used	un-used

4.2.3 Transport Firewall Rule Tree Structures Design

The design of the transport rule tree structure occurred very similarly to that of the IP rule tree. Similar to IP rules, transport rules are based on source and destination port numbers. This is particularly applicable to TCP and UDP, two of the most common protocols used on the Internet. One difference, however, is that the Internet packets are not guaranteed to utilize a specific transport protocol, if any. Therefore, a transport rule is dependent on both the port number and the protocol. Contrary to the IP rules, the port numbers and protocols used as part

of the transport rule must match exactly. This implied the use of a second rule table within the basic NP firewall, an FPM (full pattern match – see section 3.3) based tree.

The design for the transport tree memory occurred in a similar fashion to the IP rule tree. As stated, a transport rule is dependent on the specific protocol and port number. Therefore, the key used to index the transport rule tree also had to be based on these two properties. Once again, it was convenient to manage data structures on a byte-by-byte basis. Therefore, the key structure used for this tree was a concatenation of a one-byte protocol code and the two-byte port number to be filtered.

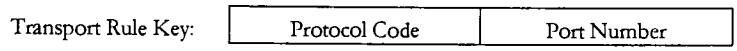


Figure 4.3 – Transport Rule Key

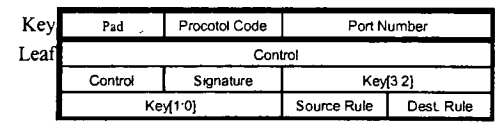
Below is an outline of the memory design process.

Important Notes:

An transport rule depends on both the protocol and port number.
Filter rules will be permitted for TCP and UDP packets
It is useful to work with the parameters on a byte by byte basis
It is a hardware requirement for the key to be a multiple of 16 bits; therefore, a pad of 1 byte is required

- The data leaf includes all information including a copy of the data used to build the search key.
- Control overhead for the leaves is 5 bytes
-

Transport Rule Leaf Structure



Key Bytes:
Field:
Pad = 0x00 - 1 Byte
Protocol Code - 1 Byte
Port Number - 2 Bytes

Leaf Data:
Field:
Control *(Used by NP hardware to manage the order of elements in the tree)*: 5 Bytes
Signature *(Identifies the Control Point that created the rule)*: 1 Byte
Key *(Copy of the key)*: 4 Bytes
Source Port Rule *(Action to perform if the key matches the protocol and source port of a packet)*: 1 Byte
Destination Port Rule *(Action to perform if the key matches the protocol and destination port of a packet)*: 1 Byte

Byte Codes for 'Source Rule' and 'Destination Rule' Fields:
Firewall Action:
Permit = 0x00
Deny = 0x01
IP Protocol Field Codes for Transport Layer Protocols:
TCP = 0x06
UDP = 0x11
*** Only TCP and UDP are supported in this design, see RFC 1700 for a full list*

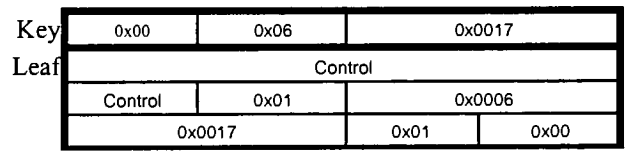
Summary:
Control Overhead - 5 Bytes
Key - 4 Bytes
Leaf Data - 8 Bytes
Total: 16 Bytes = 128 Bits

DRAM was chosen for the transport rule table as well, for the same reasoning used for the IP rule table (the size of the module). Again, the typical DRAM module is 64 bits wide. Each transport rule leaf structure, shown above, uses a total of 16 bytes or 128 bits, which implies exactly two DRAM banks are necessary. Thus, the memory allocation for transport rule table was a width of two and a height of one. Several example leaf structures are shown below.

Example Rule #1:
"Deny any packet originating from TCP port 23 (telnet)"

This implies a protocol code of 0x06 for TCP and a port number of 23 = 0x17 =>
Key: 0x00060017
Leaf Data, excluding hardware control bits (suppose Signature = 0x01): 0x01000600170100

Example Transport Rule Leaf Structure (see previous page)



This completed the memory design process for the data structures used to manage the firewall rules.

4.2.4 Test Procedure

With the completion of the firewall rule data structures, filtering code (see figure 3.) was then added to the existing router application provided by IBM, as described in section 4.1. A test network using simulated NPs was then created, and the firewall was then tested using a series of Internet operations.

4.2.4.1 Software Components

The IBM routing software consists of control point software and assembly code executed on the NP. Code additions were necessary in both. The control point was used to declare the

memory dimensions required for the IP and transport firewall rule tables. These declarations were forwarded to the NP and used to allocate the appropriate memory blocks. The control point was also the focal point for adding and manipulating the rules contained on the firewall. This was done by adding firewall options to a text based user interface provided by IBM (the npcp_console application). The firewall interface is shown on the following page.

Console Prompt:

```
>
        logout
        set
        show
        trace
        firewall
              ip
                    add   (Adds an IP filter)
                          <net mask (format aa.bb.cc.dd)> <prefix length> <source rule> <destination rule>
                    query   (Checks the values for an IP filter on the NP)
                          <net mask (format aa.bb.cc.dd)> <prefix length>
                    delete (Deletes an IP filter)
                          <net mask (format aa.bb.cc.dd)> <prefix length>
                  - update (Updates the source/destination rule of an IP filter)
                          <net mask (format aa.bb.cc.dd)> <prefix length>
                    list (Lists the statistics of IP rules in the system)


              transport
                    add (Adds a transport layer filter)
                          tcp
                                <port number> <source rule> <destination rule>
                          - udp
                                <port number> <source rule> <destination rule>
                    query (Checks the values for a transport layer filter on the NP)
                          - tcp
                                <port number>
                          - udp
                                <port number>
                  - delete (Deletes a transport layer filter)
                          tcp
                                <port number>
                          - udp
                                <port number>
                  - update (Updates the source/destination rule of a transport filter)
                          tcp
                                <port number>
                          - udp
                                <port number>
                    list (Lists the statistics of transport rules in the system)

            - load_rule_file (Loads a set of pre-defined rules from a file, using the above format used to 'add' a firewall rule)
                    <filename>  (i.e. /etc/npcp.d/firewall.conf)
```

4.2.4.2 Test Environment

The test environment for the basic NP firewall is shown on the following page. As can be seen, the test network was comprised of five systems, including one for executing the simulated NP firewall code, three hosts for performing Internet operations, one for performing network address translation (NAT) in order to hide the false internal IP addresses.

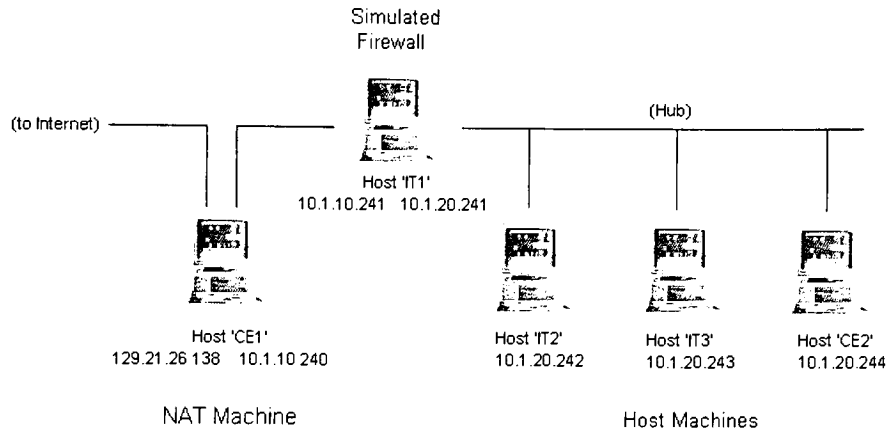


Figure 4.4 – Basic NP Firewall Test Network

Machine Name (see figure 4.4 Above)	OS	CPU	Memory
IT1	Linux 6.1	Pentium II - 300 MHz	128 MB
IT2	Linux 6.1	Pentium - 166 MHz	64 MB
IT3	Linux 6.1	Pentium - 166 MHz	64 MB
CE1	Linux 6.1	AMD K6 - 500 MHz	96 MB
CE2	Linux 6.1	Pentium - 166 MHz	64 MB

Table 4.1 – Basic NP Firewall Network Machine Descriptions

In order to test the NP, a series of Internet applications were executed from the three host machines on the network. These applications included loading a web page from a Netscape browser, performing FTP and telnet operations, and using the America Online Instant Messenger chat program. Firewall rules were added to the systems at random in order to block network traffic originating from these applications. The firewall was found to work successfully.

4.3 Parallel Packet Filter

The parallel packet filter that was implemented followed the architecture described in section 3.3. As shown in figure 3.6, this architecture is comprised of one switch NP that serves as the interface to the firewall, and two firewall NPs that share the load of filtering packets.

4.3.1 Switch Design

The switch used in this parallel firewall architecture is conceptually a straightforward entity. The switch must process all packets that travel across the firewall, and the switch must balance the network traffic load between the two firewall machines.

The first issue that arose was the matter of avoiding "circular" packet routing. According to the diagram, suppose a packet, destined for the Internet, was sent by the internal host.

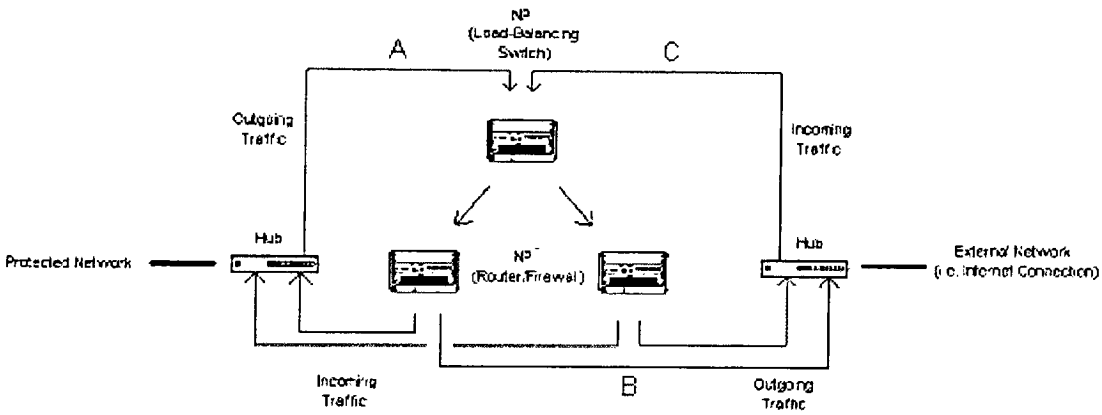


Figure 4.4 – Parallel Firewall Diagram (Highlighted)

This packet would be visible to the switch on the network at point 'A'. This packet would be forwarded by the switch to one of the firewall nodes, for example the left most firewall machine, and would appear on the external network (if permitted) at point 'B.' This (identical) packet would also be visible to the switch at point 'C' on the network, and would therefore be forwarded by the switch a second time. This process would obviously repeat until the network would become completely congested with packet traffic. The switch code, therefore, needed the ability to distinguish between the direction of the packet (direction across the firewall) and the

port at which the packet arrived. The port of arrival could easily be extracted from the control information provided by the NP hardware. The direction of the packet could be determined from the IP address information. The flowchart below outlines the algorithm performed by the NP switch code.

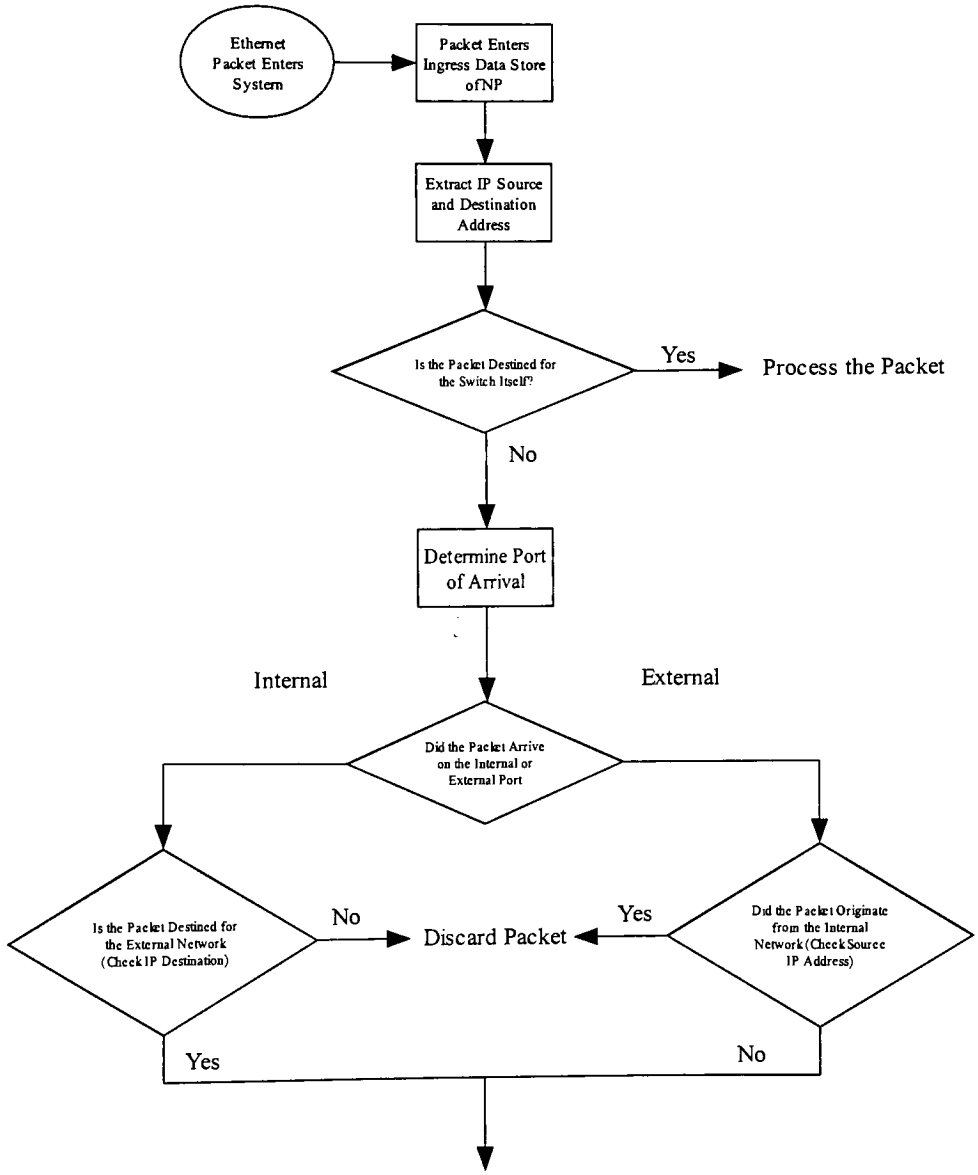


Figure 4.5 – Parallel Firewall Switch Algorithm Flowchart

The next step in the design of the switch was the algorithm for sending packets to the two firewall nodes. For this implementation a simple rotating scheme was chosen. Suppose there were 'n' firewall machines attached to the switch for the purpose of sharing the firewall load. Each firewall node would be connected to the switch via its network, 10.1.n.0, at an address of 10.1.n.242, where 242 is a randomly chosen, but fixed, value for all of the firewall nodes (see diagram below).

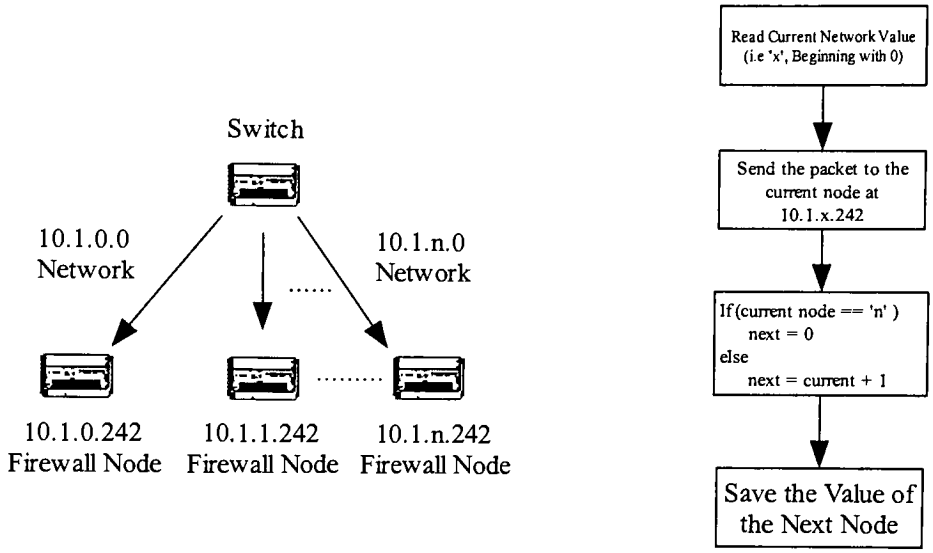


Figure 4.6 – Network Switching Scheme

The switch then rotates through each firewall node, in a round-robin fashion, sending packets for filter processing.

4.3.2 Firewall Node Design

A goal of the parallel firewall was to utilize the basic NP packet filter designed as described in section 4.2. The only additional code required was to only accept packets that enter the firewall system from the port attached to the firewall switch. This was simply a matter of checking the incoming port number, provided by the NP hardware, before continuing packet processing.

4.3.3 Test Procedure

Because the parallel firewall was intended to appear identical in behavior to the basic NP packet filter, the testing strategy was essentially the same. As done for the basic NP packet filter, this implementation was modeled using the IBM PowerNP 4GS3 simulator in a Linux environment. Below is the test network that was used.

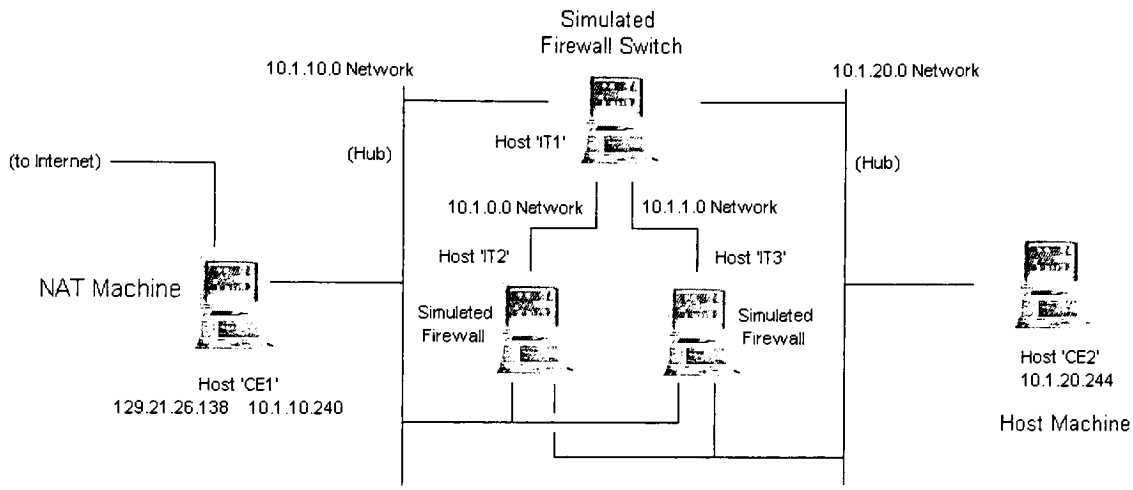


Figure 4.7 – Parallel Firewall Test Network

It can be seen from the diagram that the test network consisted of the three NP simulators already mentioned in addition to a single host used to perform various communications across the firewall and, once again, a network address translation (NAT) machine between the test network and the outside work. The systems used were the same used for the basic NP packet filter (see Table 4.1). In order to test the parallel NP, the same set of Internet applications were executed from the three host machines on the network. These applications included loading a web page from a Netscape browser, performing FTP and telnet operations, and using the America Online Instant Messenger chat program. Firewall rules were added to the firewall nodes using the user interface described in section 4.2.4.2. The following user interface was provided in order to manipulate the firewall switch.

Console Prompt:

```
>
        logout
        set
        show
        trace
        firewall
              switch
                    set
                          auto (instructs the switch use all branches)
                          assigned (instructs the switch to use a single node branch (0...n))
```

The switch was tested by running it in both the automatic and assigned branch configurations. This was done to show that the switch could be used to redirect network traffic through a specific node in the event that one of the firewall nodes went offline. All of these tests were found run successfully for the test parallel firewall network.

4.4 Results Analysis

After successfully testing the two firewall architectures in the simulated environment, it was clear that the NP provided effective means for implementing a firewall. The tree management hardware provided by the IBM PowerNP 4GS3 was necessary in order to provide an improved data model, supported by high-speed hardware, for storing firewall rules. While tree structures can support a large number of applications, the longest prefix match (LPM) and full pattern match (FPM) tree search capabilities were particularly applicable to filtering existing protocols, including IP, TCP and UDP. In addition, by utilizing the co-processors available in the NP for issuing tree memory searches, a high degree of parallelism was achieved.

While parallelism was used to improve the performance of the firewall algorithm, memory accesses still caused the most significant delay in the execution of the packet filter. Network processors typically possess a variety of memory modules, such as the DRAM modules used for storing the IP and transport firewall rules. One possible improvement that could be made to the basic packet filter architecture would be to utilize a faster memory, for example internal network processor memory. The trade-off for using the faster memory would be a constraint in the size of the firewall rule table. In many real world applications, it is possible that the firewall could grow to exceed the size of memory available internally to the network processor. One possible way to lower the size of the firewall rule table, in order to meet internal memory

constraints, would be to condense the structures used in storing the individual firewall rules. For example, many of the flags and key structures used in the IP and transport firewall rule tables for this implementation used byte codes. These rules could have been condensed to bit-sized flags in order to conserve memory. However, condensing or packing bits in this fashion would also increase the complexity of the firewall algorithm. Naturally, this would imply a slower firewall. In summary, a compromise must be found between the memory used, the structure of the firewall rules, and the algorithm for performing packet filtering.

The parallel firewall was, of course, also shown to be a feasible method for implementing a firewall using the NP. However, like the basic NP packet filter, a variety of trade-offs must be considered when implementing a parallel firewall architecture. For example, adding a switch to the parallel architecture adds a second hop for network traffic to travel. A second hop implies an added latency in executing the filtering rules. It is therefore essential that the switch mechanism, or any such packet switching algorithm that divides the filtering work among the parallel firewall machines, must provide a minimal latency with respect to the total time necessary to perform the filter rules. The IBM PowerNP 4GS3 is designed to work in conjunction with other network processors over a high-speed, custom-designed switch fabric. This type of high-speed communication media would be extremely significant in improving the performance of a parallel firewall by minimizing switching time. Another, less significant, drawback of a parallel firewall is that it requires more IP address than a standard firewall. Assume a typical firewall has one interface to the internal network and a second interface to the external network. The parallel firewall requires two interfaces at the switch, but also an addition two interfaces for each parallel firewall rule used. Thus, the number of IP addresses consumed by a parallel firewall of 'n' parallel firewall nodes would grow according to the formula below:

Equation 4.2 – Number of IP Address = $2 + 2n = O(N)$.

This is significant because the quantity of available 32-bit IP addressers available to modern industry is rapidly declining. The consumption of such IP addresses could potentially be costly to the user of the parallel firewall.

4.5 Chapter Summary

The two firewall schemes that were implemented provided a detailed look into the architectural capabilities of the IBM PowerNP 4GS3, as they related to firewall. The tree search mechanisms and co-processors provided an excellent framework for improving performance and utilizing parallelism in the implementation of packet filtering firewalls. While these mechanisms were effective, it was also clear that performance trade-offs existed for virtually all design decisions that were made. These trade-offs include compromises in memory configurations used and even the physical properties of the network. In summary, while the NP is an apparently beneficial architecture for firewall technology, the door for future exploration of and improvement to firewalls has only just been opened. The following chapter provides some additions summarizations and several ideas for future work with the IBM PowerNP 4GS3 and firewall strategies.

Chapter 5

Conclusion

5.1 Summary of Work

The work performed in this study provided a solid foundation to many of the issues involved in using a network processor as a platform for the implementation of a basic firewall system. To begin, it was apparent that the memory architecture of the network processor plays a significant role in the design of the firewall system. This was evident in the case of the IBM PowerNP 4GS3 in relation to the use of LPM and FPM tree structures for firewall rules and the necessity to balance the size of the firewall table with the capacity and speed of a particular memory module.

Parallelism was a second issue involved throughout both of the firewall implementations that were developed. Parallelism made available by packet-processors and co-processors is a definite benefit to the execution of a firewall algorithm, because of the large amount of memory accesses that typically occur for firewall rule look-ups. If parallelism can be incorporated into the rule look-up, the performance of the firewall will improve.

The implementation of the parallel firewall was significant in that it demonstrated the abilities of network processors to be used in order to create more complex firewall architectures. These more complex architectures are made possible because of the programmability of network processors, but also the speed at which network processors are able to transfer packet information between multiple systems.

In summary, the hardware assists and specialized packet processing abilities of network processors provide a wide range of possibilities for implementing firewalls. The basic NP packet filter and the parallel firewall provide a basis for more complex firewalls, including those that might implement application level filtering or stateful inspection. The performance improvements that are made available by network processors are certain to make them a preferred platform for the creation of firewall systems in the future.

5.2 Difficulties Encountered

In designing the network processor based firewall, several minor difficulties were encountered. Some of those difficulties were specific to the IBM PowerNP 4GS3 architecture. For example, the random deployment of packets to packet processors presented a challenging model for executing the appropriate code for a given packet arriving on a specific port. In other words, it would have been convenient to assign certain packet processor to execute a certain portion of code relating to packets that enter on a specific port. This would have saved the need to perform checks on the incoming ports in order to determine how the packet should be processed. Instead, these port checks slowed down the performance of the network processor. Another technical drawback encountered was a lack of semaphore operations available for use by the NP assembly code. As stated previously, the NP is comprised of multiple packet processors and co-processors, of which all could be accessing a shared memory value at any given time. The shared memory location storing the current firewall node, used by the parallel firewall switch machine, would be an example of this. This memory location could have been read and updated by any of the packet processors at any given time; therefore allowing the possibility of "write after read" errors (i.e. the second processor reads the shared value before the first processor has updated the shared value to the new current firewall node). Providing a hardware semaphore lock instruction would have simplified the process of synchronizing access to the shared memory value.

The most challenging design step was the design of the tree structures used by the basic NP firewall. This was true because it was necessary to both create efficient key structures and also provide a wide range of abilities for the packet filter. For example, the firewall has the ability to allow a filter on any existing any transport layer protocol and can filter on exact IP addresses or on varying length network addresses. Generating flexible, efficient data structures will be especially important in the design of network processor-based firewall, especially in the IBM PowerNP 4GS3.

Another constraint encountered was the compliance with existing network protocols. This was mainly evident in creating the parallel firewall. Because the switch communicated to the firewall nodes via an IP network, it was necessary to design the switching algorithm around the IP protocol rather than the communications abilities provided by the NP for inter-processor communication. For example, the proprietary switch fabric available for connecting multiple

IBM PowerNP 4GS3 processors would have been an ideal mechanism for connecting the switch to the filtering nodes of the parallel firewall.

A final difficulty encountered was the lack of documentation to describe the process of working with network processors. As network processors are a rather new technology, it is no surprise that little documentation or research examples exists. Observing more existing applications would have been beneficial. It is certain that developing firewall applications for the NP will become less difficult as more research is performed in the future.

5.3 Benefits

Network processors hope to provide a much higher bandwidth for firewall implementations. In addition to improved efficiency, the network processor will allow a greater range of packet-filtering options. This could eventually include applicability to ATM packet-filtering in addition to application level filtering achieved through observation of internal packet characteristics.

The network processor also adds a new degree of flexibility to firewall implementation. Existing firewalls and packet filtering hardware lack the ability to easily provide modification to the firewall rule. Because of the programmable nature of network processors, a new firewall implementation can quickly be applied to the target network.

Because network processors are a very new technology, it is important to obtain an understanding of the architectural advantages for firewall implementation, but also to determine the architectural shortcomings. This will lead to better use of existing network processor resources in addition to identifying new algorithms and/or techniques for packet-filter that are specific to Network Processor solutions.

5.4 Future Work

Given the architectural abilities provided by network processors, there exist many possibilities for future research in implementing firewalls using network processor technology. An excellent first step is to begin implementing higher-level firewalls: application gateways and stateful inspection firewalls. These types of firewalls would be able to further utilize the advanced capabilities of network processors, particularly the abilities to modify packet content

directly and to read into the application data of a given packet. This could allow user authentication or proxy forwarding to occur all on the network processor, avoiding the slow speeds of OS processing where these functions typically occur in current firewalls.

The basic NP packet filter and parallel firewall described throughout this document could also be enhanced to include more functionality. One useful addition to the basic packet filter would be "uni-directional" filter. Stated differently, the firewall could be upgraded to allow certain traffic in only one direction across the firewall, for example allowing users to only telnet from the inside to the outside of the protected network. The parallel firewall could be improved by automating the switch machine. When one of the firewall machines goes offline, it would be used for the switch to detect the crash, and re-route packets to a different firewall node appropriately.

Given the firewall's dependencies on memory for accessing firewall rules, it would be highly beneficial to perform a more detailed analysis of the effects produced by using different memory modules. This would obviously be most useful after upgrading the test environment to use actual network processor hardware, rather than simulated systems.

A final recommendation for further study is to pursue the advantages of using special purpose switch fabrics to interconnect network processor-based firewalls. As described for the parallel firewall, using an Ethernet-based network for inter-system communication would ultimately deteriorate the performance of the firewall. A switch fabric between the network processors would greatly reduce the latency involved in data transfer between network processor systems.

Firewalls have become one of the basic building blocks of implementing a network's security policy; and therefore, should be made as efficient as possible. Because a firewall can potentially lead to a bottleneck in the network, improving the performance of the firewall means also improving the performance of the protected network. With their specialized packet processing abilities and hardware support functions, network processors represent a new generation of higher-speed, more affective firewalls.

BIBLIOGRAPHY

1. Ashley, Park and Mark Vandenwauver. *Practical Intranet Security: Overview of the State of the Art and Available Technologies.* Kluwer Academic Publishers, 1999.

2. Bace, Rebecca Gurley. *Intrusion Detection.* Macmillan Technical Publishing, 2000.

3. Benecke, Carsten. *A Parallel Packet Screen for High Speed Networks.* Universitat Hamburg. Proceedings of the DARPA Information Survivability Conference and Exposition Volume I. IEEE, 1998.

4. Black, Uyless D. *Internet Architecture: An Introduction to IP protocols.* Prentice Hall PTR, 2000.

5. Chess, David. *The Future of Viruses on the Internet.* Virus Bulletin International Conference, San Francisco, California, October 1-3, 1997.

6. Crowley, Patrick, Marc E. Fiuczynski, Jean-Loup Baer, and Brian N. Bershad. *Characterizing Processor Architectures for Programmable Network Interfaces.* 2000 International Conference on Supercomputing, Sante Fe, N.M., May, 2000.

7. Goncalves, Marcus. *Firewalls: A Complete Guide.* The McGraw-Hill Companies, Inc., 2000.

8. Gordon, Sarah. *Virus Writers – The End of the Innocence?* Virus Bulletin Conference, September, 2000.

9. Hazelhurst, Scott, Adi Attar, Raymond Sinnappan. *Algorithms for Improving the Dependability of Firewall and Filter Rule Lists.* University of the Witwatersrand. Proceeding of the International Conference on Dependable Systems and Networks (DSN 2000). IEEE, 2000.

10. Huang, Michael E. *Virtual Private Networks – Real-world Challenges and Solutions.* Internet Asia Magazine, April 1, 2000.

11. Hursh, Matt. *Firewalls for Beginners.* Zteck Inc. November 6, 2000.

12. IBM Microelectronics. *The Network Processor: Enabling Technology for High-Performance Networking.* International Business Machines, 1999.

13. Kephart, Jeffrey O., Gregory B. Sorkin, David M. Chess, and Steve R. White. *Fighting Computer Viruses.* IBM Thomas J. Watson Research Center. Yorktown Heights, NY.

14. Ker, Keith. *Internet Firewalls: Questions and Answers.* Proceeding of the International Society for Optical Engineers, Information Protection and Network Security Conference, October 24-25, 1995.

15. Kurose, James F. and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet.* Addison Wesley Longman, Inc., 2001.

16. Lakshman, T.V. and D. Stiliadis. *High-Speed Policy-Based Packet Forwarding Using Efficient Multi-dimensional Range Matching.* Bell Laboratories, February 8, 1998.

17. *Securing Your Network with the Cisco Centri Firewall.* Chapter 3: "Evolution of the Firewall Industry" and Chapter 4: "Inside the Cisco Centri Firewall." Cisco Systems, Inc., 1997.

18. Schuba, Christop Ludwig. *On The Modeling, Design, and Implementation of Firewall Technology.* Purdue University. December, 1997.

19. Siyan, Karanji and Chris Hare. *Internet Firewalls and Network Security.* New Riders Publishing, 1995.

20. Spatscheck, Oliver, Jorgen S. Hansen, John H. Hartman, and Larry L. Peterson. *Optimizing TCP Forwarder Performance.* IEEE/ACM Transactions on Networking, Vol. 8, No. 2, April 2000.

21. Stateful Inspection Technology. Check Point Software Technologies, Ltd., 1999.

22. Stevens, W. Richard and Gary R. Wright. *TCP/IP Illustrated, Volume 2: The Implementation.* Addison-Wesley, 1995.

23. Tanenbaum, Andrew S. *Computer Networks – Third Edition.* Prentice Hall PTR, 1996.

24. White, Steve R. *Open Problems in Computer Virus Research.* IBM Thomas J. Watson Research Center. Yorktown Heights, NY. Presented at Virus Bulletin Conference, Munich, Germany, October 1998.

25. Xu, Jun and Mukesh Singhal. Design of a High-Performance ATM Firewall. 5th Conference on Computer and Communications Security. ACM 1998.

26. Zwicky, Elizabeth D., Simon Cooper, and D. Brent Chapman. *Building Internet Firewalls.* O'Reilly & Associates, Inc., June 2000.

APPENDIX A – FIREWALL VENDORS

1. Advanced Network and Services, Inc. (InterLock Firewall) – http://ww.ans.com
2. AltaVista Software (Firewall 98 – The Active Firewall)
3. AXENT (Raptor) – http://www.raptor.com or http://www.axent.com
4. Cisco Systems (PIX Firewall) – www.cisco.com
5. Check Point Software Technologies (Firewall 1) – http://www.checkpoint.com
6. CyberGuard (CyberGuard Firewall) – http://www.cybg.com
7. CYCON (Labyrinth) – http://www.cycon.com
8. Global Technology (GNAT Box) – http://www.gnatbox.com
9. MilkyWay (SecurIT FIREWALL) – http://milkyhway.com
10. NetGuard Ltd. (Guardian) – http://www.netguard.com
11. Network-1 Software and Technology (CyberWall Plus, FireWall Plus) – http://www.network-1.com
12. Secure Computing (Sidewinder, Borderware Firewall) – http://www.securecomputing.com
13. Sun Microsystems (SunScreen Firewall Suite, Solstice FireWall) – http://www.sun.com
14. Technologic (Interceptor Firewall) – http://ww.tlogic.com
15. Trusted Information Systems (Gauntlet Internet Firewall) – http://www.tis.com
16. WatchGuard Technologies (Firebox) – http://www.watchguard.com

APPENDIX B – NETWORK PROCESSOR VENDORS

1. Agere Systems (NPFPP, NPRSP) – http://www.agere.com
2. Chrysalis-ITS (Luna 340) – http://www.chrysalisits.com
3. Conexant Systems (MXT4400, CX27510) – http://www.conexant.com
4. EZ Chip Technologies (NP-1) – http://www.ezchip.com
5. Fast-Chip Incorporated (PolicyEdge) – http://www.fast-chip.com
6. Intel Corporation (IXP1200, IXP220, IXP225) – http://www.intel.com
7. International Business Machines (NP4GS3, NPr2.7, NPe405x) – http://www.chips.ibm.com
8. Internet Machines – http://www.internetmachines.com
9. Lara Networks (NAP Family) – http://www.laranetworks.com
10. Lexra (LX8000, NetVortex OC192) – http://www.lexra.com
11. MMC Networks (EPIF, XPIF, GPIF) – http://www.mmcnetworks.com
12. Motorola (C5DCP) – http://www.e-motorola.com
13. SiberCore Technologies (SiberCAM Ultra-2M) – http://www.sibercore.com
14. Silicon Access Networks (iFlow Family) – http://www.siliconaccess.com
15. Vitesse Semiconductor Corporation (OctoalMAC,IQ2000) – http://www.vitesse.com
16. ZettaCom Incorporated (ZEN, ZET Families) – http://www.zettacom.com